Thesis
Reports
Albini,
F. A.

FINAL REPORT FOR RESEARCH GRANT
#INT-92754-GR
Montana State University
FS Contact: James K. Brown
Co-op Contact: Frank Albini

PROGRAM BURNUP

A SIMULATION MODEL OF THE BURNING OF LARGE WOODY NATURAL FUELS

30 July 1994

Frank A. Albini
Adjunct Professor
Mechanical Engineering Department
Montana State University
Bozeman MT 59717

Final report on Research Grant INT-92754-GR

by

USDA Forest Service, Intermountain Research Station

to

Montana State University Mechanical Engineering Department

CONTENTS

I. FUNCTIONAL DESCRIPTION OF PROGRAM OPERATION

II. HOW TO USE THE MODEL

III. DETAILED DESCRIPTION OF THE OPERATION OF EACH MODEL ELEMENT

APPENDIX A. BURNING RATE CALIBRATION

1. SELECTION OF VALUES FOR BURNING RATE CONTROLLING PARAMETERS

2. EXAMPLES OF MODEL PERFORMANCE IN PREDICTING CRIB BURN RATES

APPENDIX B. SOURCE CODE LISTING

OVERVIEW OF PROGRAM BURNUP

The BURNUP program serves as a "driver" that calls in the required order the routines that gather the data, organize it, process it, and report the results. The program follows this logical sequence:

1) Call GETDAT. This routine accepts direction from the user via keyboard entry to read data files or to accept keyboard input that describes the fuels and the environmental variables, prompts desired modifications of these data, and facilitates their archiving.

2) Query user whether to store fire history or not. This prompt also serves to mark return of control to the driver program from GETDAT.

3) Call ARRAYS. This routine reorders the list of fuel components on the basis of diameter, moisture content, and mass density, in order to facilitate the assignment of the fractions of loadings of each size class whose burning rates are influenced by the same and smaller sized elements. It then computes the planform projected area of each class of fuel elements and assigns the interaction fractions to an ordered list representing a triangular matrix.

4) Initialize the time marker to the duration of the igniting fire, and the output snapshot index to unity.

5) Call DUFBRN. This routine establishes the rate at which duff is estimated to burn, based on its means moisture content, and the time for which it will burn, based on the mean duff loading, the fraction estimated to burn (based on mean moisture also) and the burning rate.

6) Call START. This routine exposes the fuel array to the intensity of the igniting fire for the period of time specified. The status of each component of the fuel complex is established at the end of this time as being burned out, burning, or not yet ignited, and quantities describing these states are calculated, such as the burning rates for each of the pairs of interacting fuel elements, the diameter and the loading of each element remaining, the heat absorbed up to present time for fuel elements not yet ignited, overall fire intensity, etc. If no fuel is ignited START terminates the run with a screen message.

7) Assign duff fire intensity if duff still burning.

8) If fire history record is desired, call STASH. This routine will create and file a snapshot of the status of each fuel element when it is called. When the routine is first called, user specifies whether the history is to include a detailed record of each interacting pair of fuel elements or only a summary. The summary gives the gross and dry weight total fuel loading, the gross and dry loading fractions remaining, and the fire intensity and, of course, the time.

9) Test fire intensity level. If not greater than the minimum value of 100 W/m^2, end computation by branching to 17 below.

Data Requirements

The user must provide a description of the fuel complex to be burned and the conditions under which the fire occurs, by giving values for the set of parameters that describe the fuel bed, the environment, and each distinct fuel component. In addition, some program-control parameters must be specified. The parameters that must be quantified are:

Fuel component data

1. Identity (*e.g.*, '0 - 1/4 in twig')
2. Quantity per unit area (kg / m^2 , ovendry)
3. Surface area / volume ratio (m^{-1})
4. Moisture content (fraction dry mass)
5. Heat of combustion (J / kg)
6. Total mineral content (fraction dry mass)
7. Mass density (kg / m^3 , ovendry)
8. Ignition temperature (deg C)
9. Characteristic pyrolysis (char) temperature (deg C)
10. Thermal conductivity, ovendry (W / m K)
11. Specific heat capacity, ovendry (J / kg K)

Environmental data and ignition pulse description

1. Ambient temperature (deg C)
2. Fuelbed depth (m)
3. Windspeed at top of fuelbed (m / s)
4. Intensity of ignition pulse (kW / m^2)
5. Duration of ignition pulse (s)

Empirical and program control parameters

1. Fire environment temperature* parameter r0
2. Fire environment temperature* parameter dr
3. Fuel planform interaction area factor** K_a
4. Time interval per step (s)
5. Maximum number of time steps to simulate

Program management flags

1. Is a fire history file to be created? (T , F)
2. Is a detailed element-interaction history desired? (T , F)
3. Is a detailed element-interaction summary desired? (T , F)

---

* Fire environment temperature is derived from current area intensity of fire, I, duff burning intensity, I_d, and interaction with other burning fuel components that are the same size or smaller. Quantities r0 and dr quantifiy the influence of fuel component interaction (see description of subroutine STEP).

** The "effective" planform area of a fuel component, for the purpose of determining its level of interaction with other fuel components, is given by K_a times its physical planform area.

SUBROUTINE GETDAT

This subroutine facilitates the input, review, revision, and archiving of data needed to operate the BURNUP code package. It is entirely menu driven and organized in a hierarchy of increasingly detailed choices. Operation of this routine is perhaps easiest to comprehend through a discussion of the sequence of menues it displays. GETDAT keeps count of the number of times it has been called, which measures the number of exercises done during one session of using the BURNUP program. On the first call to this routine all numerical data are initialized to zero and the names of all fuel components is set to 'no data for'. On subsequent calls, initialization is skipped and the routine immediately displays the first menu (prompts are *shown in italic*, responses underlined):

1.0 First menu:

Enter 1 for keyboard entry of data
Enter 2 to fetch data stored in files
Enter 0 to terminate program now Enter KEY

If KEY = 0 STOP
If KEY = 2 go to 2.0
If KEY = 1, keyboard entry selected, these prompts are given, in order:

1.1 *How many [no more than MAXNO] fuel components?* Enter NUMB

Set i = 0
1.1.0 i = i + 1
1.1.1 *Fuel component number i properties list*
Name [12 characters or fewer] Enter name(i)
1.1.2 *Name(i) dry loading, kg / sq m* Enter wdry(i)
1.1.3 *Name(i) mineral ash fraction* Enter ash(i)
1.1.4 *Name(i) heat of combustion, J / kg* Enter htval(i)
1.1.5 *Name(i) moisture content, fraction* Enter fmois(i)
1.1.6 *Name(i) ovendry mass density, kg /cubic m* Enter dendry(i)
1.1.7 *Name(i) surface / volume ratio , inverse m* Enter sigma(i)
1.1.8 *Name(i) specific heat capacity ovendry J/kg K* Enter cheat(i)
1.1.9 *Name(i) ovendry thermal conductivity W / m K* Enter condry(i)
1.1.10 *Name(i) ignition temperature deg C* Enter tpig(i)
1.1.11 *Name(i) char [end pyrolysis] temperature deg C* Enter tchar(i)
If i < NUMB go to 1.1.0

If this was a revision of data, go to 1.2

1.1.12 *Enter igniting surface fire intensity, kW/sq m* Enter fi
1.1.13 *Enter igniting surface fire residence time, s* Enter ti
1.1.14 *Windspeed at top of fuelbed, m / s* Enter u
1.1.15 *Depth of fuelbed, m* Enter d
1.1.16 *Ambient temperature, deg C* Enter tpamb
1.1.17 *Dimensionless area influence factor [ak param]* Enter ak
1.1.18 *Fire environment minimum temperature parameter r0* Enter r0

1.1.19 *Fire environment increment temperature param dr* Enter dr
1.1.20 *Time step for integration of burning rates , s* Enter dt
1.1.21 *Number time steps [must exceed 1]* Enter NTIMES
1.1.22 *Duff dry weight loading, kg / sq m* Enter wdf
1.1.23 *Duff moisture fraction* Enter dfm

1.2 Review / revise / archive menu display:

All data can be reviewed and revised now. Enter:
0 to archive and/or execute present data
1 to review fuel component properties
2 to review igniting fire and environmental data
3 to review internal and control variables
4 to terminate program now Enter NEXT

if NEXT = 4 STOP
if NEXT = 3 go to 1.5
if NEXT = 2 go to 1.4
if NEXT = 0 go to 3.0
if NEXT = 1 then:

1.2.1 *Enter 0 to terminate review & revision of fuel data*
Enter 1 to select a fuel component for review
Enter 2 to add or delete a fuel component Enter IDO

if IDO = 0 go to 1.2
if IDO = 2 go to 1.3

1.2.2 List all fuel components by name and loading:

Name(i) , wdry(i) for i = 1 to NUMB
Number component to be revised [0 = prev menu] Enter nn

if nn = 0 go to 1.2.1

1.2.3 *index parameter data entered*

1	*name*	*Name(nn)*
2	*loading*	*wdry(nn)*
3	*ash frac*	*ash(nn)*
4	*heat comb*	*htval(nn)*
5	*mois frac*	*fmois(nn)*
6	*density*	*dendry(nn)*
7	*surf/vol*	*sigma(nn)*
8	*heat capy*	*cheat(nn)*
9	*conduct y*	*condry(nn)*
10	*ig temp C*	*tpig(nn)*
11	*char temp*	*tchar(nn)*

enter number param to change [0 = prev menu] Enter idn

if idn = 0 go to 1.2.2

if idn = 1:

1.2.3.1 *Enter new name - 12 char or fewer* Enter Name(nn)

if 1 < idn < 12:

1.2.3.2 *New value = ?* Enter new value for selected param

go to 1.2.3

1.3 Add or delete fuel component menu display:

Enter - 1 to delete a fuel component
Enter 0 if no more additions or deletions
Enter + 1 to add a fuel component Enter IDO

if IDO = 0 go to 1.2

if IDO = - 1 then:

1.3.1 List all fuel components by name and loading:

Name(i) , wdry(i) for i = 1 to NUMB
Index number of component to delete Enter ind

Delete chosen entry, NUMB = NUMB - 1, go to 1.3

if IDO = + 1 then:

i = NUMB, NUMB = NUMB + 1, go to 1.1.0

1.4 Review igniting fire and environmental data menu:

1 - Igniting fire intensity, kW / sq m *fi*
2 - Igniting fire residence time, s *ti*
3 - windspeed at top of fuelbed, m / s *u*
4 - fuelbed depth, m *d*
5 - ambient temperature, deg C *tpamb*
Index of parameter to change [0 = prev menu] Enter ind

if ind = 0 go to 1.2
revise value of chosen parameter, go to 1.4

1.5 Review internal and control variables menu:

1 - Area influence factor [ak parameter] *ak*
2 - Fire environment temperature parameter r0 *r0*
3 - Fire environment temperature parameter dr *dr*
4 - Time step to integrate burning rate , s *dt*
5 - Duff dry weight loading , kg / sq m *wdf*
6 - Duff moisture content, fraction dry weight *dfm*

7 - Number time steps *NTIMES*
Index of parameter to change [0 = prev menu] Enter ind

if ind = 0, go to 1.2
revise value of chosen parameter, go to 1.5

2.0 Fetch data from files

2.1 Fuel component data file:

Enter name of file holding fuel component data Enter INFILE

open, read, and close INFILE to establish fuel component data

2.2 Igniting fire, environmental, and control variables file:

Enter name of file holding igniting fire, environmental, and program control parameters Enter INFILE

open, read, and close INFILE to establish these data

2.3 go to 1.2

3.0 Archive / execute menu:

Enter 2 for previous menu
Enter 1 to archive current data set
Enter 0 to execute without archiving Enter IDO

if IDO = 0 RETURN FROM THIS ROUTINE TO PROGRAM BURNUP
if IDO = 2 go to 1.2
if IDO = 1 then:

name, open, and write data files
1) fuel component properties data
2) igniting fire, environmental, and control data
go to 3.0

Appended to subroutine GETDAT is the utility routine RETRY that compares the values of quantities input from keyboard against prestored "small" and "large" values of the parameter quantified. If the value lies outside of the anticipated range, the input is rejected and the user is prompted to enter the quantity again. This utility is intended to prevent pathological cases arising when the user enters data from a list and one quantity is either skipped or entered twice, causing nonsensical data entry. This test routine can readily be defeated if desired.

SUBROUTINE ARRAYS

This subroutine manages the following tasks:

1) It sees to the ordering of the list of fuel elements so that their internal enumeration corresponds to

Increasing diameter, with ties broken by
increasing moisture content, with ties broken by
increasing mass density (ties not resolved).

2) Using the ordered list of fuel elements, it partitions the loading of each size class into a secondary list of loadings, each representing the loading that interacts with other specific elements. It also assigns the initial diameter of each entry in this secondary list.

3) It returns an array (variable name: AREA) whose elements are the fractions of the burn site plan area expected to be covered at least once by the planform projected area of each fuel element. These fractions are used later in establishing the local fire intensity levels for each interacting pair of fuel elements.

The first task is executed by subroutine SORTER, which uses the properties of surface area / volume ratio (sigma), moisture content (fmois), and dry mass density (dendry) to sort the fuel element list. It performs an in-situ sort on these three parameters and returns the list order in the array named KEY. This integer array is used by ARRAYS to reorder all the other fuel properties. Doing this sort first greatly simplifies expression of the logic of the fuel element interaction algorithm.

The second and third tasks are expedited by a call to subroutine OVLAPS, which returns the variables AREA and ALONE as well as the triangular matrix ELAM. Element j of the array ALONE represents the fraction of the loading of fuel #j that does not interact with any other fuel elements of the same or smaller size class (elements with indices less than or equal to j). These "ALONE" elements are influenced by duff burning, however.

Matrix element ELAM(j,k) represents the fraction of the loading of fuel element j whose burning will be influenced by fuel element k. In computing this fraction, OVLAPS uses the assumption (which persists from the first attempt at large fuel burnout modeling [Albini 1976]) that the strength of the interaction is proportional to the ratio of the diameters of the two interacting fuel elements. The matrix is triangular:

$$ELAM(j, k > j) = 0.$$

This can (improperly) be viewed as the assumption that the burning of fuel element k > j will not influence the burning of element j. But the correct interpretation of this equation is that the interaction between elements j and k should be computed only once, and this is done by considering each size class as being influenced by its own and smaller size classes. Matrix ELAM and vector ALONE are merged in the array XMAT, to allow overlaying to shrink the RAM required by the program if that becomes necessary.

SUBROUTINE DUFBRN

This brief subroutine is appended to the BURNUP driver module file. It combines the following elements to make a prediction of the rate of duff burning and the duration of burning of duff:

The rate of smoldering consumption of moist peat is given by Frandsen [1991A] as:

$$dw''/dt = 0.27 - 0.097\ R_M - 0.033\ (\ R_I - D\ )\ ,\ gm\ /\ h\ cm^2 \qquad (1)$$

where R_M is the ratio of moisture mass to dry organic mass, R_I the ratio of inorganic mass to dry organic mass, and $D = R_I$ for $R_I < 1.0$; here we use

$$dw''/dt = 7.5 \times 10^{-4} - 2.7 \times 10^{-4}\ R_M\ ,\ kg\ /\ s\ m^2\ . \qquad (2)$$

The heat evolved from smoldering peat was measured by Frandsen [1991B] at about 15 MJ / kg, varying only slightly with moisture content, imperceptibly with inorganic content, and not at all with bulk density. Multiplying the mass consumption rate by this figure gives a duff fire intensity I_d:

$$I_d = 11.25 - 4.05\ R_M \quad kW\ /\ m^2\ . \qquad (3)$$

Now if the duff loading is W_d (kg / m^2), we can use the correlation of Brown et al [1985] for the fractional duff depth reduction, F, to compute a burning duration, t_d, as:

$$F = 0.837 - 0.426\ R_M \qquad (4)$$

$$t_d = F\ W_d\ /\ (\ 7.5 \times 10^{-4} - 2.7 \times 10^{-4}\ R_M\ ) \qquad (5)$$

These equations are implemented by subroutine DUFBRN.

SUBROUTINE START

Subroutine START performs the tasks necessary to begin repetitive calls to subroutine STEP, which simulates the time history for each interacting pair of fuel components and so builds the fire intensity history. The preliminary routine, START, establishes the conditions obtaining at the end of the ignition pulse and establishes values for the following event times for the "larger" [see ordering rules in SORTER] component of each interacting pair of fuel elements:

1) Time of the start of drying of the surface (achievement of 80 C)

2) Time of ignition

3) Time of burnout (complete consumption, diameter = 0)

In addition to these times, the following quantities are cumulated for each of these fuel elements:

4) Cumulative heat input per unit surface area

5) Time integral of difference between fire environment temperature and surface temperature [used in time-to-drying-start estimate]

6) Time integral of the square of the ratio of fuel dry thermal conductivity to effective film heat transfer coefficient since time of ignition [used in establishing time over which to average in calculating burning rate]

The following time dependent simulation quantities are sampled (at the end of the ignition pulse) and stored for each of these fuel elements:

7) Post-ignition history of heating rate

8) Diameter reduction rate

9) Dry loading loss rate

Finally, the fraction of each fuel component loading that is currently burning and the fraction that has burned out are cumulated by summing over all the inferior-size interaction partners for each of these fuel elements. The order in which the calculations are performed is:

1) Initialize time varying quantities and store the ratio of diameter reduction rate to surface heating rate for each component.

2) Estimate start of drying time for each component.

3) Determine which components are lit by ignition pulse; reset time origin to minimum ignition time.

4) Establish burning rate for each ignited component and extrapolate to end of ignition pulse.

SUBROUTINE FIRINT

This utility subroutine returns the value of the fire intensity, kW/m^2, for the burn site as a whole, and correction terms to be added to it in order to estimate the local level of fire intensity in the "interaction area" of each fuel size class.

To calculate the burn site average intensity, the routine sums (over the inferior order elements) the load loss rate for each interaction pair, to determine the site-average net loading loss rate for each size fuel component. This quantity is adjusted for mineral content and multiplied by the heat of combustion for that size class to obtain its contribution to the site average fire intensity..

Simultaneously, the site-average intensity contribution of each size class is divided by the fraction of the site area expected to be covered by the effective planform area of the size class (which determines its contribution to fire intensity where these elements are actually located) and the site-average contribution is subtracted. The resulting quantity then represents a correction term which can be added to the site-average fire intensity to determine the intensity to be expected in the near vicinity of a fuel element of this size class, in order to establish a local fire environment characterizing interactions with this size class.

SUBROUTINE STASH

When this routine is called, it writes to a designated formatted file the current values of the quantities: time, remaining total gross loading, remaining dry weight loading, remaining fraction of initial gross loading, remaining fraction of initial dry loading, and fire intensity.

If the user has specified to do so, the routine first stores the following quantities for each fuel component (say component number K): time, fraction of loading currently lit, and fraction of loading burned out. In addition, the following data are output for each of its interaction partners, in order, from j = 0 (no partner) to j = K (same size class):

1) Time, sec

2) Dry weight loading interacting with size class j

3) Rate of dry loading loss due to interaction with size class j

4) Current diameter of size K elements interacting with size class j

5) Diameter reduction rate of these size K elements

6) Time when these elements reached surface drying state

7) Time when these elements ignited

8) Time when these elements burned out

SUBROUTINE TIGNIT

This routine solves the inverse problem of finding the time at which the temperature of the radiantly heated surface of a halfspace achieves a certain temperature. The temperature of the surface as a function of time can be expressed in terms of the dimensionless temperature rise as a function of dimensionless time. The characteristic time for this heating problem is given by the product of the thermal conductivity and the heat capacity per unit volume for the heated halfspace, divided by the square of the effective film heat transfer coefficient [see subroutine HEATX]. The solution is expressed in terms of a dimensionless temperature ratio given by the difference between the fire environment temperature [see TEMPF] and the component ignition temperature divided by the difference between the fire environment temperature and the ambient temperature. Solving for the dimensionless time at which this surface temperature is achieved involves seeking a root of a an approximate equation for the complementary error function [Hastings 1955]. The predicted ignition time is returned in sec, and the empirically determined correction factor of 0.5 is applied in the calling routine.

SUBROUTINE DRYTIM

The time at which the surface of a radiantly heated halfspace achieves the temperature at which moisture loss begins is found by this routine. The result is expressed in terms of the Fourier number (time multiplied by fuel thermal diffusivity and divided by square of fuel element radius) which is the quantity actually returned. This is achieved by virtue of supplying the routine with the Biot number (film heat transfer coefficient times diameter divided by fuel thermal conductivity) and the temperature rise target in the form of the ratio of the surface temperature rise at the start of drying to the difference between the fire environment tamperature [see TEMPF] and the ambient temperature. The solution is achieved by binary search, using an approximate expression for the complementary error function [Hastings 1955].

SUBROUTINE HEATX

This routine characterizes the heat transfer between the fire environment and a fuel element. Inputs to this routine are horizontal windspeed at top of the fuelbed, fuelbed depth d, fuel particle diameter, fire environment temperature, fuel surface temperature, and fuel thermal conductivity in moist state. The convective film heat transfer coefficient is computed from the Nusselt number, predicted from $0.344Re^{0.56}$, where Reynolds number Re is computed for air at fire environment temperature moving laterally over the fuel particle at a speed given by the vector sum of the horizontal velocity at the top of the fuelbed and vertical velocity estimated roughly as $(0.53\ gd)^{0.5}$. A radiation contribution is added to give an effective film coefficient, and moist state conductivity is used with diameter to yield a Biot number. Output are convective and effective film heat transfer coeffients and the net effective Biot number.

FUNCTION TEMPF

This function returns the parameter described here as the "fire environment temperature", T_F, which quantity represents the temperature to which an inert small object would rise if it were to reside with the fire at a specified location indefinitely. It is presumed that this temperature can be calculated from the local fire intensity, I, (in kW/m^2), the ambient temperature, T_a, and a single parameter, r, which nominally characterizes the amount of ambient air entrained within the local flame fluid flow. The rationale behind the formula for the parameter T_F is very simple. We posit a heat flow balance at the top of the fuelbed in the form:

$$\dot{m}'' C_p (T_F - T_a) + \varepsilon\sigma (T_F^4 - T_a^4) = I$$

where

$\dot{m}''$ = combustion product fluid flux ($kg/s\ m^2$)
C_p = fluid specific heat capacity ($kJ/kg\ K$)
ε = area averaged radiometric emissivity
σ = Stefan-Boltzmann constant
= (5.67×10^{-11} $kW / m^2\ K^4$)

The intensity, I, we presume to be proportional to the total rate of fuel loading reduction, m_f'', which, in turn, we assume to be proportional to $\dot{m}''$ so we have

$$I = H\ m_f''$$

with H representing a mean heat of combustion, and

$$\dot{m}'' = (1 + \Phi)\ m_f''.$$

This leads finally to the form

$$\vartheta = 1 + r / (1 + A\ r (\vartheta + 1)(\vartheta^2 + 1) / I)$$

where

$$\theta = T_F / T_a$$

$$r = H / (1 + \Phi)C_p\ T_a$$

$$A = \varepsilon\sigma\ T_a^4 .$$

The maximum value of r is estimated to be about 2.5, being largest when two flaming components are interacting and smallest when a fuel element burns in isolation. The parameter A is about 0.4 kW/m^2 if ε is unity; considering the fact that only a fraction of the area emits radiation at the level represented by T_F and that the emissivity of radiating zones is less than unity, an arbitrarily fixed value of 0.05 is used in the TEMPF algorithm. The algorithm returns the value of T_F using input values of I, r, and T_a, solving the equation for ϑ by iteration.

SUBROUTINE STEP

This workhorse subroutine is the heart of the simulation. It is called once for each time step. For all its apparent complexity, the logic of the algorithm is actually rather simple. It goes through the list of fuel components in increasing order of "size" (see ordering criteria in SORTER), and for each element (say =K) it examines all inferior "sizes" (say j) from j = 0, representing size K components not interacting with any other component, to j = K, representing size K components interacting with other size K components. This examination sequence is used to "update" the diameter, diameter reduction rate, loading, loading reduction rate, cumulative heat input, etc., for each subcategory. The logic of the sequence of examinations is verbalized in the following paragraphs.

1) For each subcategory of components of size class K that has not yet burned out, the routine determines if the subcategory will burn out by the end of the present time step. If so, the diameter reduction rate and loading loss rate are approximated by their current values divided by the estimated time duration until burnout.

2) If it will not burn out by the end of the time step, but is already burning, then values of r and local fire intensity I_K, for use in subroutine TEMPF are assigned by the following rules:

i) if j = 0, then
$r = r0 + 0.5\ dr$
$I_K = I + I_d$

ii) if $0 < j < K$, then
$r = r0 + 0.5\ (1 + f_j)\ dr$
$I_K = I + \Delta I_K + f_j \Delta I_j$

iii) if j = K, then
$r = r0 + 0.5\ (1 + f_K)\ dr$
$I_K = I + f_K \Delta I_K$

In these equations it is recommended that the empirical parameters r0 and dr take on the values 1.8 and 0.4 respectively [see appendix on burning rate calibration]; f_j is the fraction of size class j that is presently burning, and ΔI_j is the "local fire intensity correction term" described under subroutine FIRINT.

3) With r and local fire intensity I_K determined, the locally applicable fire environment temperature T_F is established [see TEMPF] and the rate of heating established by a call to HEATX with fuel surface temperature set to the char temperature for fuel component K. The time history of this rate, back to a maximum of MXSTEP time steps prior to the present, is updated and maintained for each subcategory of fuel component K.

4) Using this (possibly) truncated time history of the heating rate of this subcategory, a recent average heating rate is determined by selecting the minimum of three time spans over which to perform the averaging:

i) Time elapsed since ignition of this subcategory of component K

ii) The ratio of the time integral of $(c_K / h_{ejK})^2$, with c_K the dry state thermal conductivity of fuel component K and h_{ejK} the effective film heat transfer coefficient, to the thermal diffusivity (dry state) of fuel component K. The integral gives a measure of the square of the distance in from the fuel surface that the heat pulse has penetrated to the present time.

iii) The square of the current radius of this subcategory of fuel component K divided by its dry state thermal diffusivity, a measure of the time required for heat input at the surface to reach the center of a cylindrical fuel element.

Using the minimum of the above three times, the recent average heating rate is established and used to compute the diameter reduction rate and loading loss rate for the subcategory.

5) If the subcategory is not yet burning but its outer surface has reached the drying temperature, the time until ignition is predicted using an average heat transfer rate given by the ratio of the cumulative heat input to the components of this subcategory (per unit surface area) to the time integral of the temperature difference driving the heat input. A call to subroutine TIGNIT and multiplication by 0.5 gives the predicted ignition time.

If the subcategory is predicted to ignite before the end of the current time step, the current heating rate (once ignited) is used to establish a diameter reduction rate, loading loss rate, and burnout time; necessary extrapolations are done to give the state of the subcategory components at the end of the current time step.

6) If the subcategory is not yet ignited and its surface has not yet risen to drying temperature, the time when this will occur is predicted using the following sequence of computations:

i) A fire environment temperature is determined, using rules akin to those outlined in step 2) above for fixing r and I_K.

ii) The time integral of the temperature difference between the fire environment and the surface is cumulated, along with the heat transferred to the fuel components of this subcategory.

iii) The ratio of the cumulative heat transfer and the time integral of the driving temperature difference is used to establish an average film heat transfer coefficient, and the time average of the driving temperature difference to establish a dimensionless temperature rise for a call to DRYTIM.

If this computation predicts that the fuels of this subcategory will reach the surface drying state before the end of the current time step, then the ignition time is predicted for it as well. If the predicted ignition time is before the end of the current time step, then an initial entry is made in the list of samples making up the running recent history of the heating rate of the subcategory, but a diameter reduction rate is not computed.

7) Finally, a second sweep through all the subcategories of fuel components is performed, to compute the fraction of each size class burning at the end of the current time step, and the fraction that will have burned out. These fractions are based upon the initial loadings of the size classes, not the current values.

SUBROUTINE SUMMARY

When an exercise of the simulation is complete, this subroutine is called to write into a specified formatted file a summary of the initial and end states of the fire.

The initial state is described by specifying the area factor (effective / actual planform area ratio, or effective diameter factor) used for this simulation run and the final time in seconds, followed by a list of each of the surface fuel components comprising the site loading. This list gives the name of each component, its initial dry loading, its moisture content, and its initial diameter.

The minimal final state summary, which is created by default, consists of a list of the fuel components by name, and for each component, its initial loading (dry weight), its first ignition time (minimum ignition time of all subcategories), its last burnout time (maximum burnout time of all subcategories), and the loading remaining at the end time.

If the user chooses, the final state summary is elaborated in more detail. This is done by listing, for each fuel component, instead of the quantities above, its name, initial loading (dry weight), moisture content, and diameter, followed by a list of the inferior components with which this component category interacted. This list includes the following detailed information:

1) Inferior component name

2) Fraction of loading of the larger component in this subcategory

3) Ignition time for this subcategory

4) Burnout time for this subcategory

5) Final dry weight loading in this subcategory

6) Final diameter in this subcategory

This subroutine will not automatically overwrite an existing summary file, but prompts the user that this will be done if desired. This feature is present because the calling program, BURNUP, supplies the file name to this subroutine, and on repeated exercises, this name remains the same. The user may specify an alternative file name to avoid an overwrite.

REFERENCES

Albini, F. A., 1976: Computer-based models of wildland fire behavior: A users' manual. USDA For. Serv. Intermountain Research Station, Ogden UT Unnumbered publ. 68 pp.

Anderson, Hal E., 1990: Relationship of fuel size and spacing to combustion characteristics of laboratory fuel cribs. USDA For. Serv. Res. Pap. INT-424, Intermountain Research Station, Ogden UT. 9 pp.

Brown, J. K., M. A. Marsden, K. C. Ryan, and E. D. Reinhardt, 1985: Predicting duff and woody fuel consumption in the northern rocky mountains. USDA For. Serv. Res. Pap. INT-337. 23 pp.

Frandsen, William H., 1991A: Burning rate of smoldering peat. Northwest Science, Vol 65 No 4, pp 166-172.

Frandsen, William H., 1991B: Heat evolved from smoldering peat. Internatl. J. Wildland Fire Vol 1 No 3, pp 197-204.

Hastings, Cecil, Jr. 1955: Approximations for digital computers. Princeton University Press, 201 pp. See pp 167-169.

APPENDIX A. BURNING RATE CALIBRATION

To establish suitable values for the empirical parameters r0 and dr described in subroutine STEP, the model was used to simulate the burning of laboratory cribs constructed of specially dimensioned wooden elements.

These crib burns, carried out under controlled conditions in the burning chamber at the USDA Forest Service's Intermountain Fire Sciences Laboratory in Missoula Montana [Anderson 1990] were done to establish the influence of fuel component size and spacing on the rate of combustion of the crib. If the fuel elements are spaced relatively far apart, there are ample passageways for air to enter and combustion products to exit the crib structure. In this case, the rate of combustion is expected to be controlled by fuel moisture content and the total exposed surface area of the fuel elements. But if the fuel elements are packed tightly together, the restricted area at the top of the crib through which the combustion products formed in the interior of the crib must exit can become the rate-controlling parameter for burning if fuel moisture content is not.

To explore and define the transition between these two rate-controlling mechanisms, cribs were continuously weighed as they were burned and the rates of weight loss, normalized by initial weight, compared. Cribs were constructed so that their burning rates would definitely be controlled by one or the other mechanism, and some that would have the mechanisms competing for control of the burning rate.

As a result, many of these cribs were extremely heavily loaded in comparison to natural fuel accumulations. In nature, fuels are seldom, if ever, so compactly arranged that the burning rate of an accumulation would be controlled by ventillation area. Thus the model, working properly, should be expected to reproduce well the burning rate histories of loosely packed (fuel surface area controlled) cribs and to predict too rapid burning of the tightly packed (ventillation area controlled) cribs.

The following tables and graphs illustrate the performance of the model in predicting the burning rates of these laboratory crib fires. The search for appropriate values of the parameters r0 and dr was not exhaustive, but a reasonably wide range of values was explored. This was done by exercising the model for a selected set of experiments, one or two from each of four categories:

1) Rate controlled by fuel exposed surface area
2) Rate controlled by top surface ventillation area
3) Rate control mechanism uncertain (1 or 2 above)
4) Rate control mechanism not determined in documentation

The model predictions (unconnected square symbols on the following graphs) of the fraction of initial load remaining as a function of time were compared graphically with measurements (solid lines) for the set of burns. Then a different pair of parameters r0 and dr was selected, keeping the sum r0 + dr -- and hence the maximum fire environment temperature -- constant, and the process repeated. With the best visual comparison was achieved, the whole process was repeated for a different sum r0 + dr until a final selection was achieved.

The tables and graphs following are organized as follows:

1) A table giving the burn id number and the dry weight loading of each size class for all 34 burns reported by Anderson [1990].

2) A list of burns with rate controlled by exposed fuel surface area, giving initial dry weight load (kg) and species of principal fuel.

   Following this table are graphs comparing predicted and observed fraction of initial gross weight - versus - time. For the first crib, results are shown for three pairs of parameters r0 and dr. For the next two cribs, two choices are shown. For the remainder, only the recommended values r0 = 1.8, dr = 0.4 are shown.

3) A list of burns with rate controlled -

   i) by top surface ventillation area (6 burns)
   ii) by top surface ventillation or fuel surface area (6 burns)
   iii) by undetermined mechanism (5 burns)

   The initial dry weight load is given for each crib, along with the species of the principal fuel component.

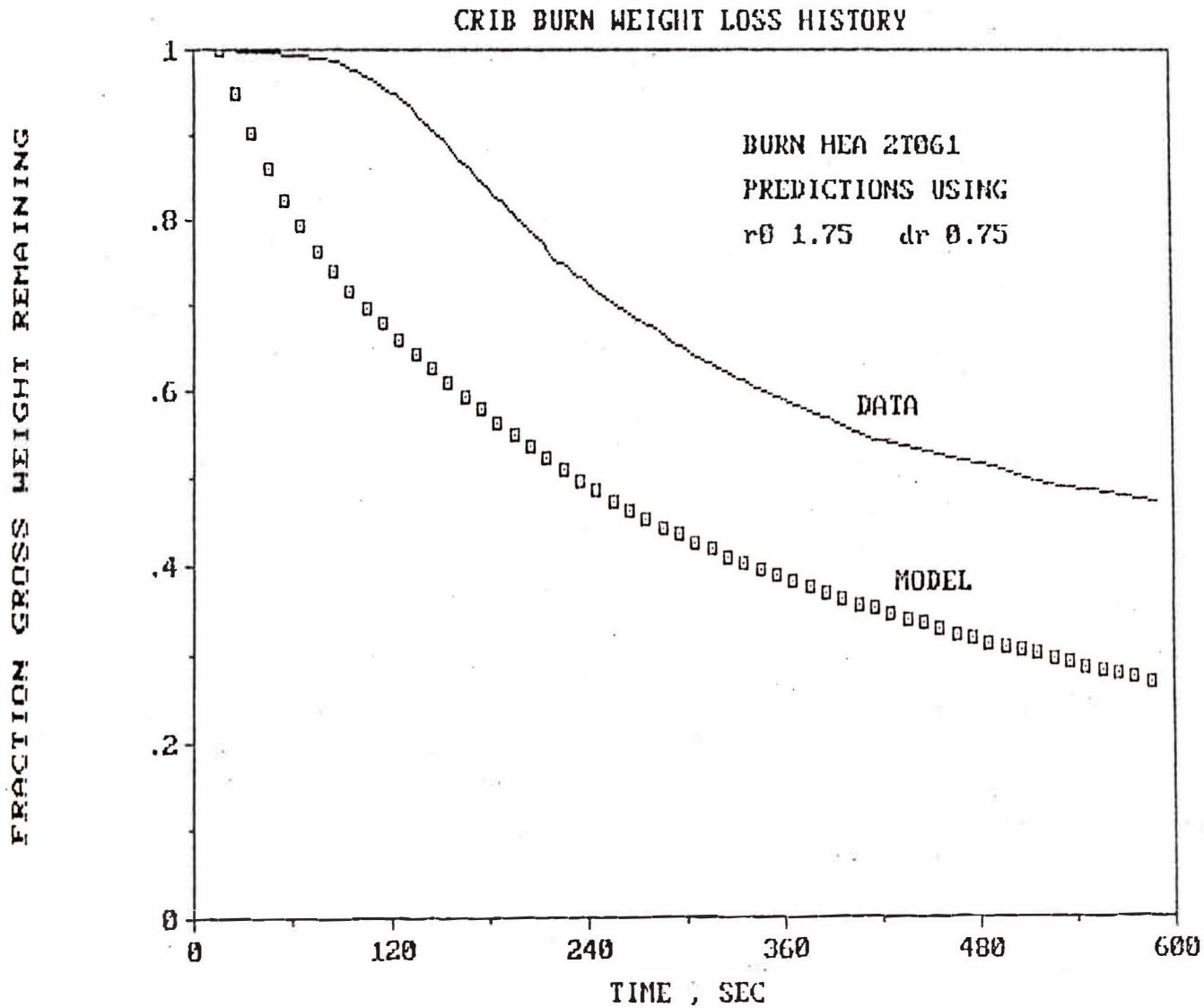
   Following this table are graphs showing predicted and observed fraction of initial gross weight - versus - time.

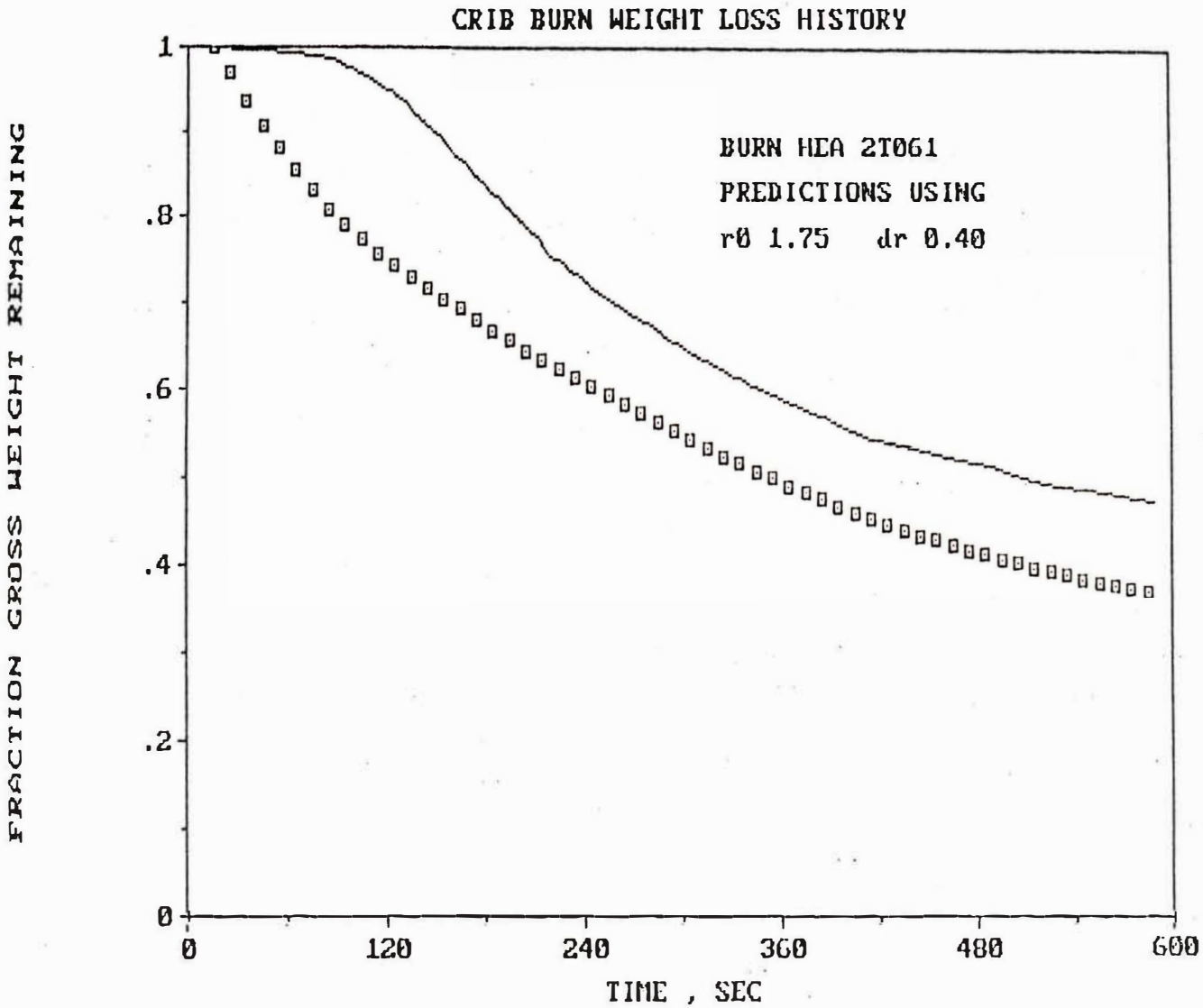
4) A series of graphs showing model predictions of intensity as a function of time for a sampling of cribs from each category. The intensities exhibit a rapid initial rise to a maximum, followed by a decline with time that varies from case to case. The decay rate obviously is not strongly correlated with the total fuel loading, nor is the maximum intensity. These graphs illustrate the sensitivity of the model to properties of the fuel accumulation other than fuel quantity.

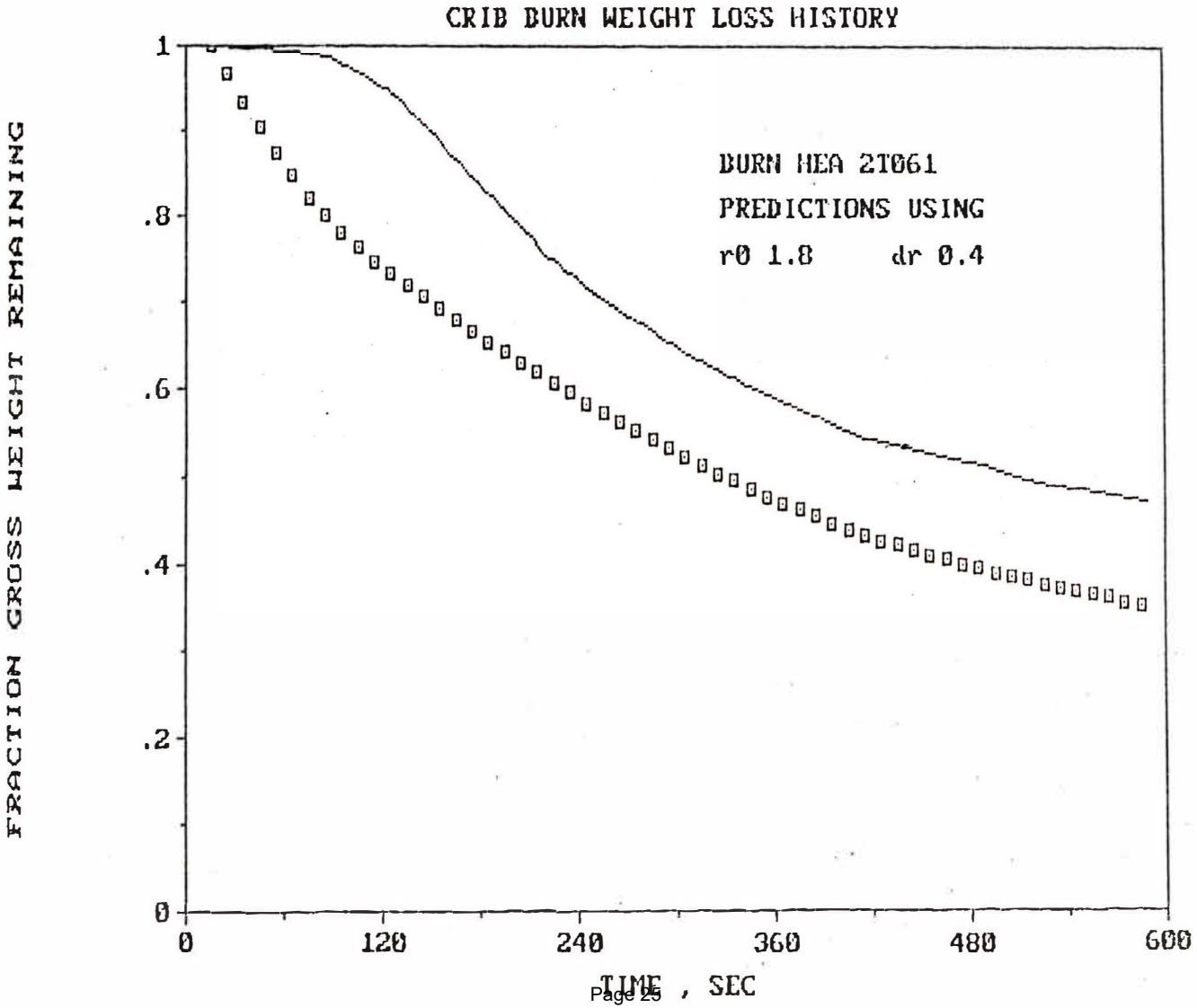
BURN NUMBER	DRY LOADING , kg / sq m					
	EXCELSIOR	1/4 INCH	1.0 INCH	2.0 INCH	4.0 INCH	6.0 INCH
HEA1T014	.537E+00	.430E+01	.141E+02			
HEA1T015	.644E+00	.413E+01	.124E+02			
HEA2T061	.732E+00	.119E+02	.309E+02		.636E+02	
HEA2T111	.513E+00	.479E+01	.135E+02	.339E+02		
HEA3T131	.668E+00	.489E+01	.141E+02			.115E+03
HEA3T141	.632E+00	.427E+01	.125E+02			.141E+03
HEA3T151	.586E+00	.421E+01	.123E+02		.110E+03	
HEA3T161	.305E+00	.214E+01	.854E+01	.436E+02		
HEA3T171	.134E+01	.381E+01	.110E+02			
HEA3T172	.134E+01	.381E+01	.139E+02			
HEA3T173	.134E+01	.381E+01	.120E+02			
HEADF21	.275E+00	.162E+01	.561E+01	.311E+02		
HEADF31	.166E+00	.173E+01	.601E+01	.515E+02		
HEADF41	.532E+00	.286E+01	.897E+01		.634E+02	
HEADF51	.397E+00	.318E+01	.994E+01		.109E+03	
HEADF52	.397E+00	.318E+01	.994E+01		.109E+03	
HEALP21	.214E+00	.128E+01	.433E+01	.239E+02		
HEALP31	.191E+00	.127E+01	.430E+01	.302E+02		
HEALP41	.532E+00	.286E+01	.897E+01		.521E+02	
HEALP51	.436E+00	.291E+01	.908E+01		.668E+02	
HEALP61	.121E+00	.158E+01	.544E+01	.601E+02		
HEAPP101	.488E+00	.244E+01	.772E+01			.610E+02
HEAPP21	.214E+00	.131E+01	.455E+01	.251E+02		
HEAPP31	.133E+00	.130E+01	.455E+01	.390E+02		
HEAPP41	.239E+00	.136E+01	.478E+01	.218E+02		
HEAPP51	.465E+00	.256E+01	.974E+01		.570E+02	
HEAPP61	.363E+00	.251E+01	.777E+01		.711E+02	
HEAPP71	.763E+00	.363E+01	.111E+02			.931E+02
HEAPP81	.430E+00	.348E+01	.106E+02			.130E+03
HEAPP91	.326E+00	.339E+01	.101E+02			.146E+03
HEASF21	.214E+00	.128E+01	.449E+01	.248E+02		
HEASF31	.118E+00	.132E+01	.457E+01	.515E+02		
HEASF41	.483E+00	.294E+01	.936E+01		.570E+02	
HEASF51	.206E+00	.242E+01	.760E+01		.107E+03	

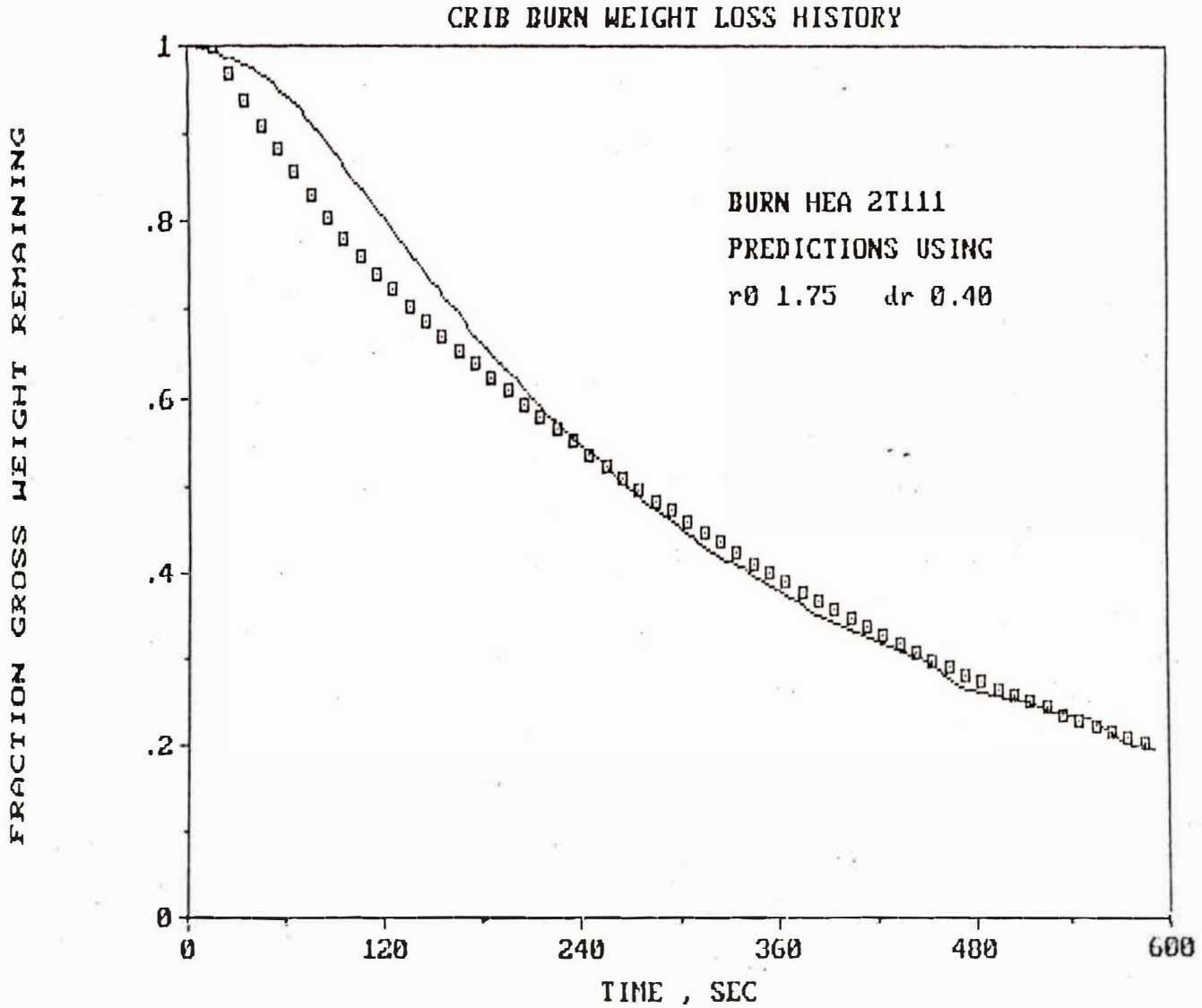
CRIB BURNS WITH RATE CONTROLLED BY EXPOSED FUEL SURFACE AREA

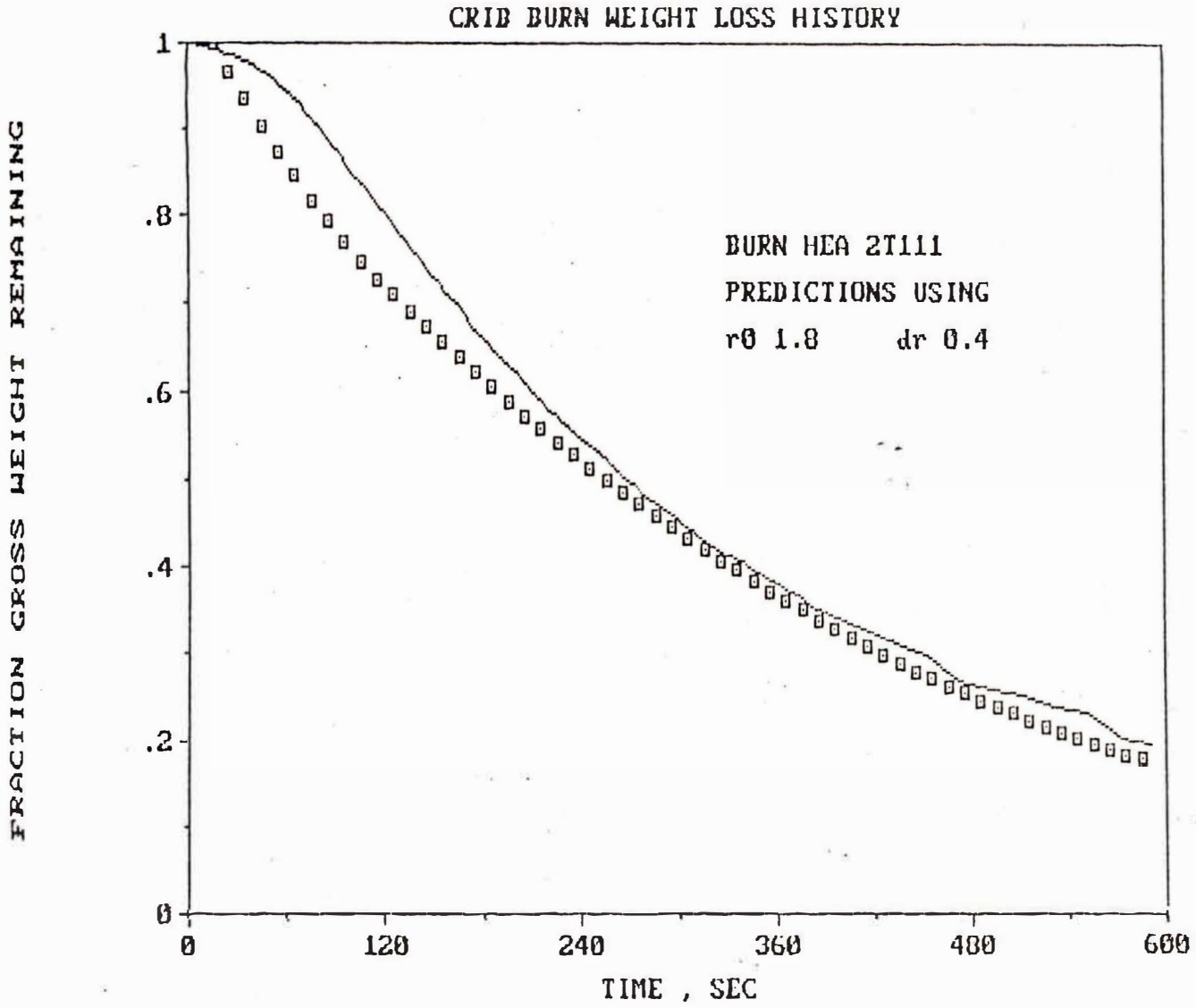
BURN ID	Dry Weight, kg	Principal fuel
HEA2T061	159.3	Ponderosa pine
HEA2T111	78.3	Ponderosa pine
HEA3T131	199.6	Ponderosa pine
HEADF21	57.4	Douglas fir
HEADF41	103.4	Douglas fir
HEALP21	44.1	Lodgepole pine
HEALP31	51.3	Lodgepole pine
HEALP41	32.1	Lodgepole pine
HEALP51	98.9	Lodgepole pine
HEAPP101	106.5	Ponderosa pine
HEAPP31	61.4	Ponderosa pine
HEAPP41	32.1	Ponderosa pine
HEAPP61	102.1	Ponderosa pine
HEAPP71	161.5	Ponderosa pine
HEASF21	45.7	Spruce / fir
HEASF41	72.0	Spruce / fir

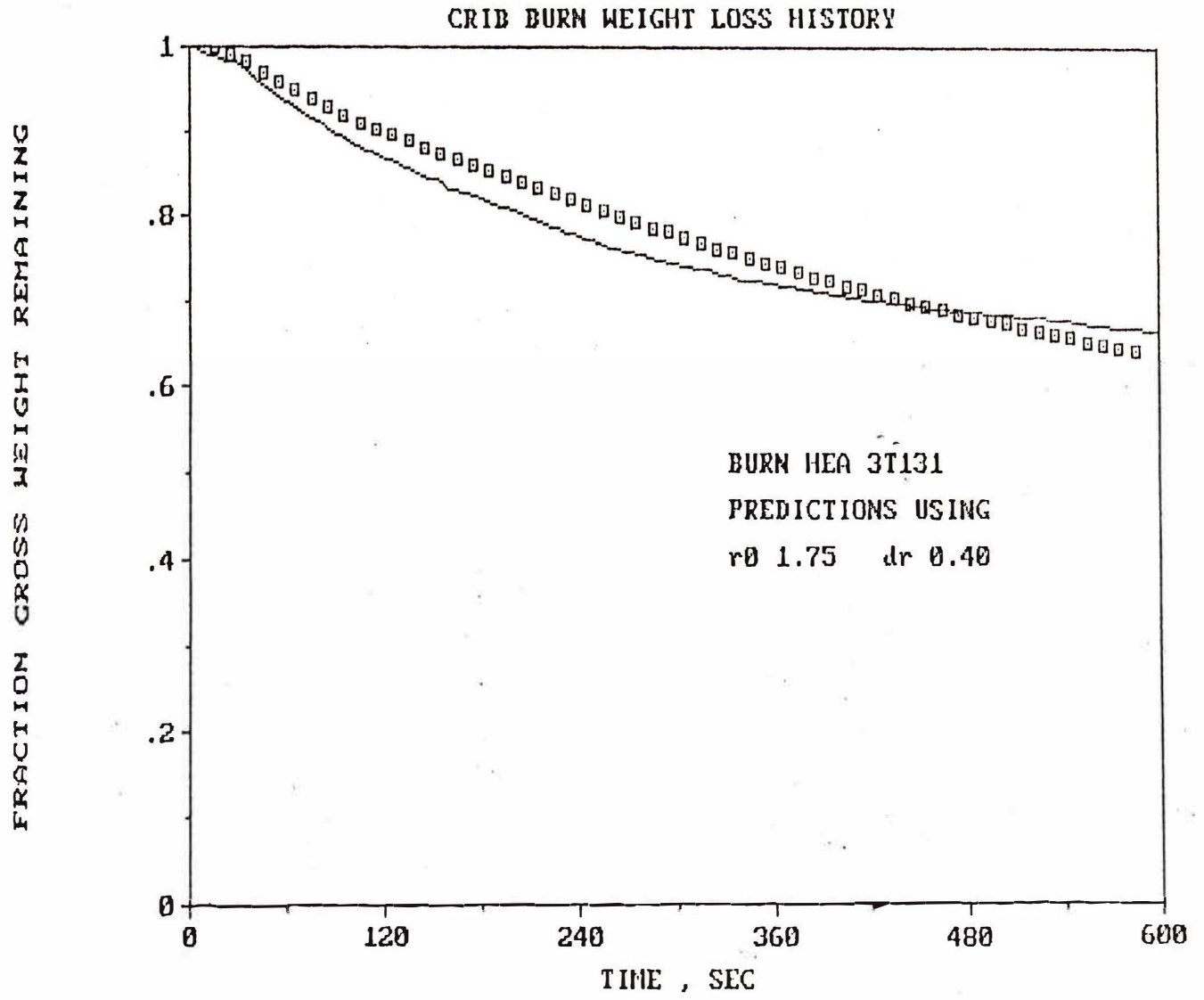

CRIB BURN WEIGHT LOSS HISTORY
BURN HEA 2T061
PREDICTIONS USING
r0 1.75 dr 0.75
DATA
MODEL
FRACTION GROSS WEIGHT REMAINING
1
.8
.6
.4
.2
0
0
120
240
360
480
600
TIME , SEC

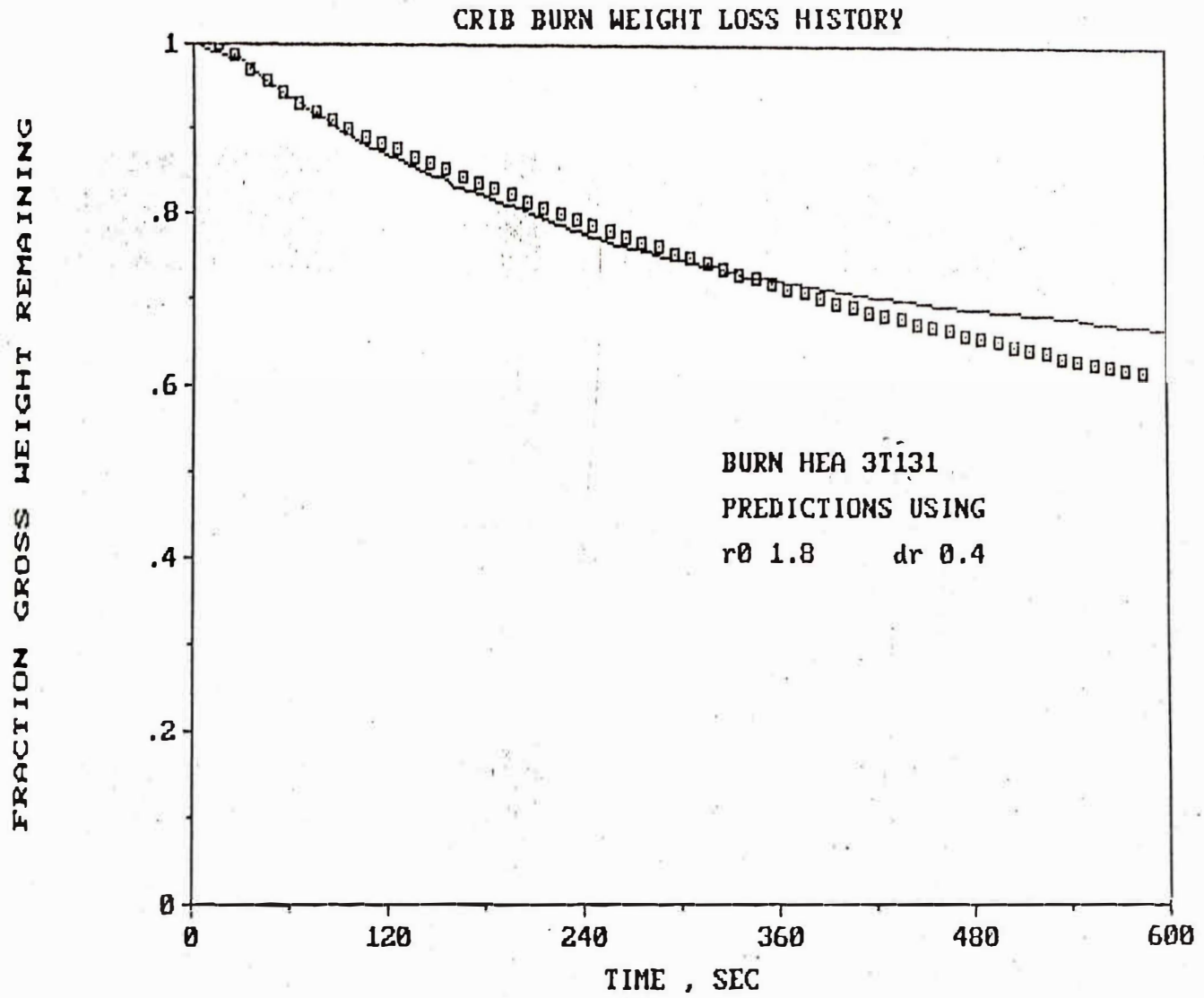

CRIB BURN WEIGHT LOSS HISTORY
FRACTION GROSS WEIGHT REMAINING
1
.8
.6
.4
.2
0
0
120
240
360
480
600
TIME , SEC
BURN HEA 2T061
PREDICTIONS USING
r0 1.75 dr 0.40

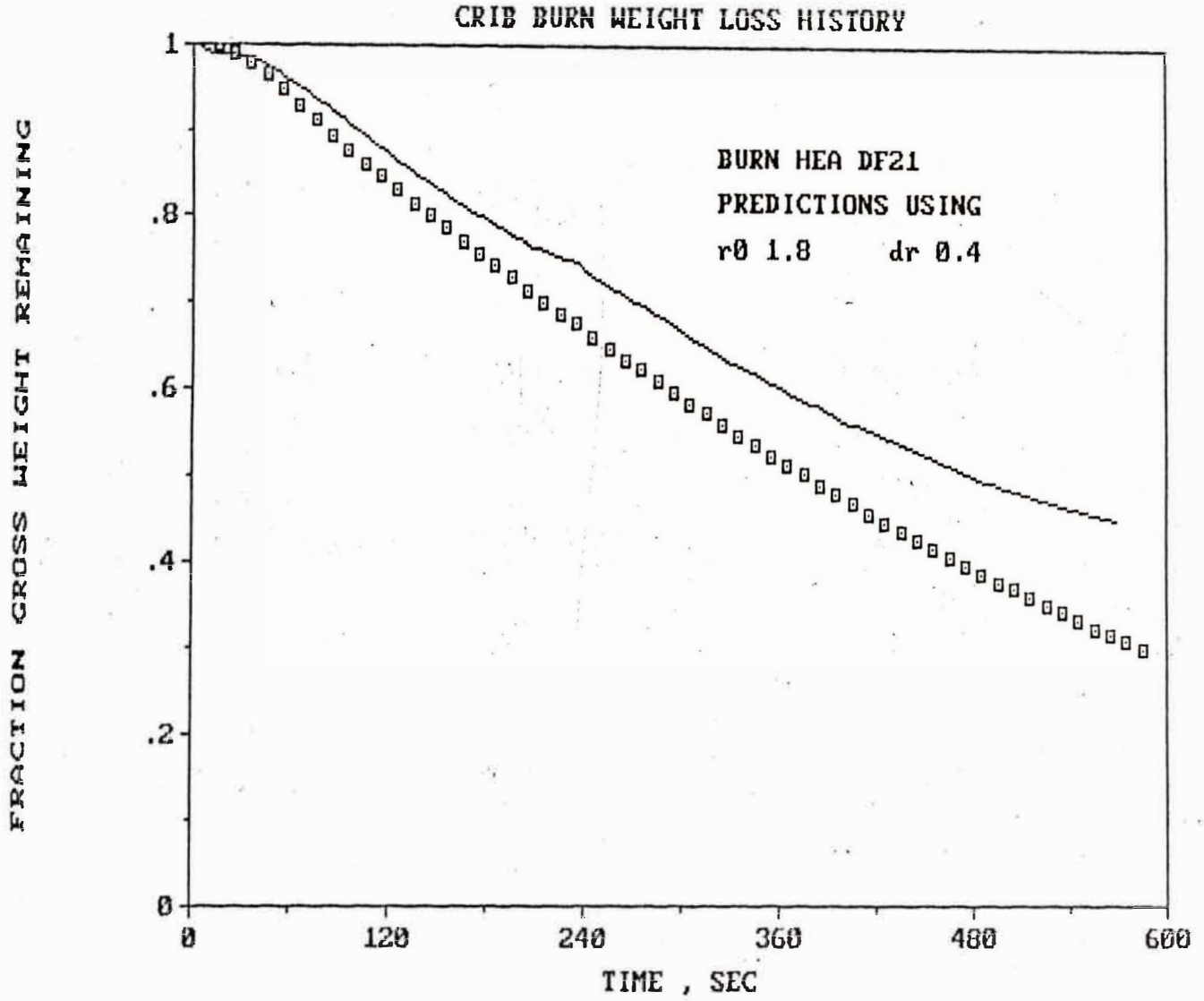

CRIB BURN WEIGHT LOSS HISTORY
BURN HEA 21061
PREDICTIONS USING
r0 1.8 dr 0.4
FRACTION GROSS WEIGHT REMAINING
1
.8
.6
.4
.2
0
0
120
240
360
480
600
TIME , SEC

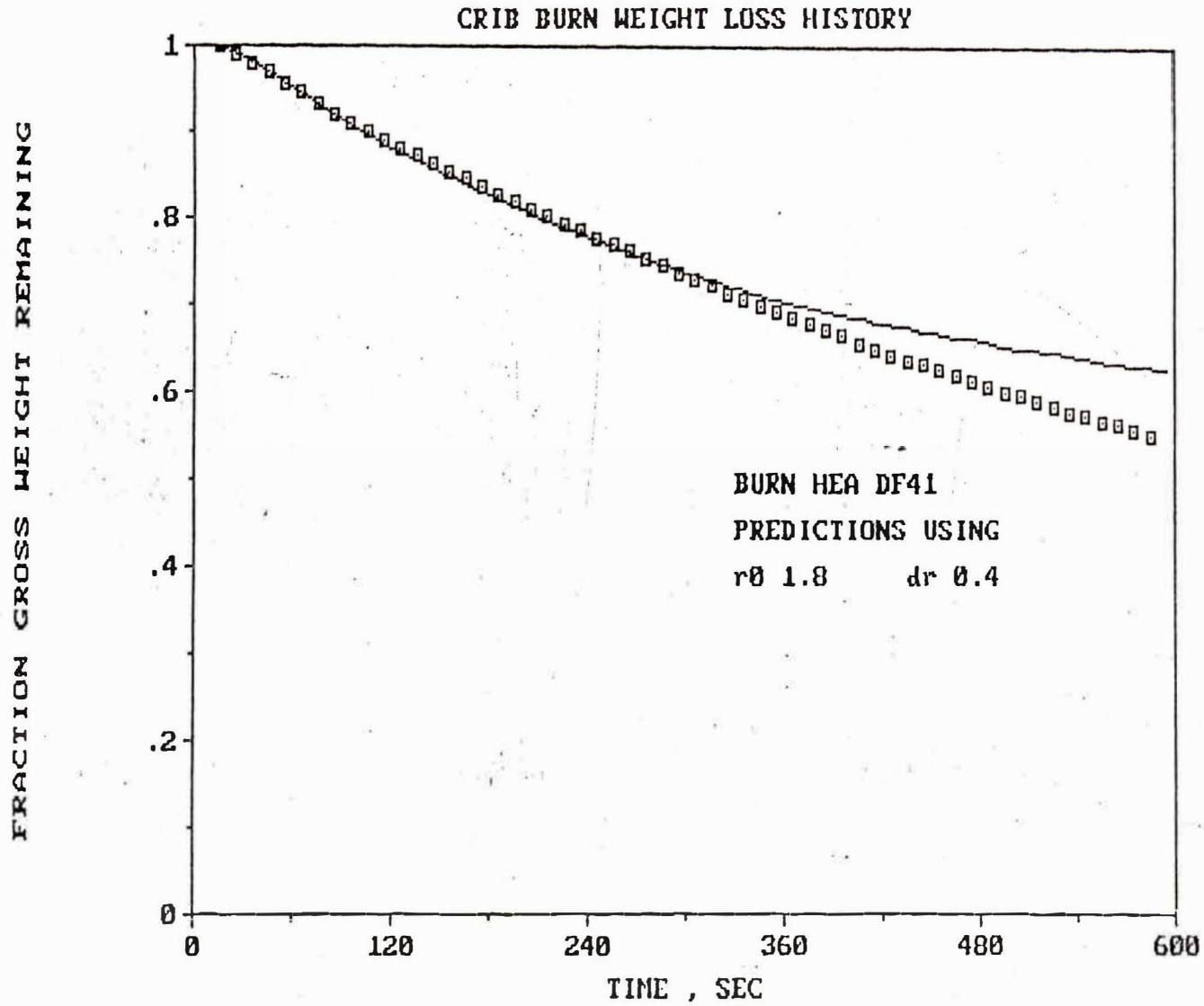

CRIB BURN WEIGHT LOSS HISTORY
FRACTION GROSS WEIGHT REMAINING
1
.8
.6
.4
.2
0
0
120
240
360
480
600
TIME , SEC
BURN HEA 2T111
PREDICTIONS USING
r0 1.75 dr 0.40

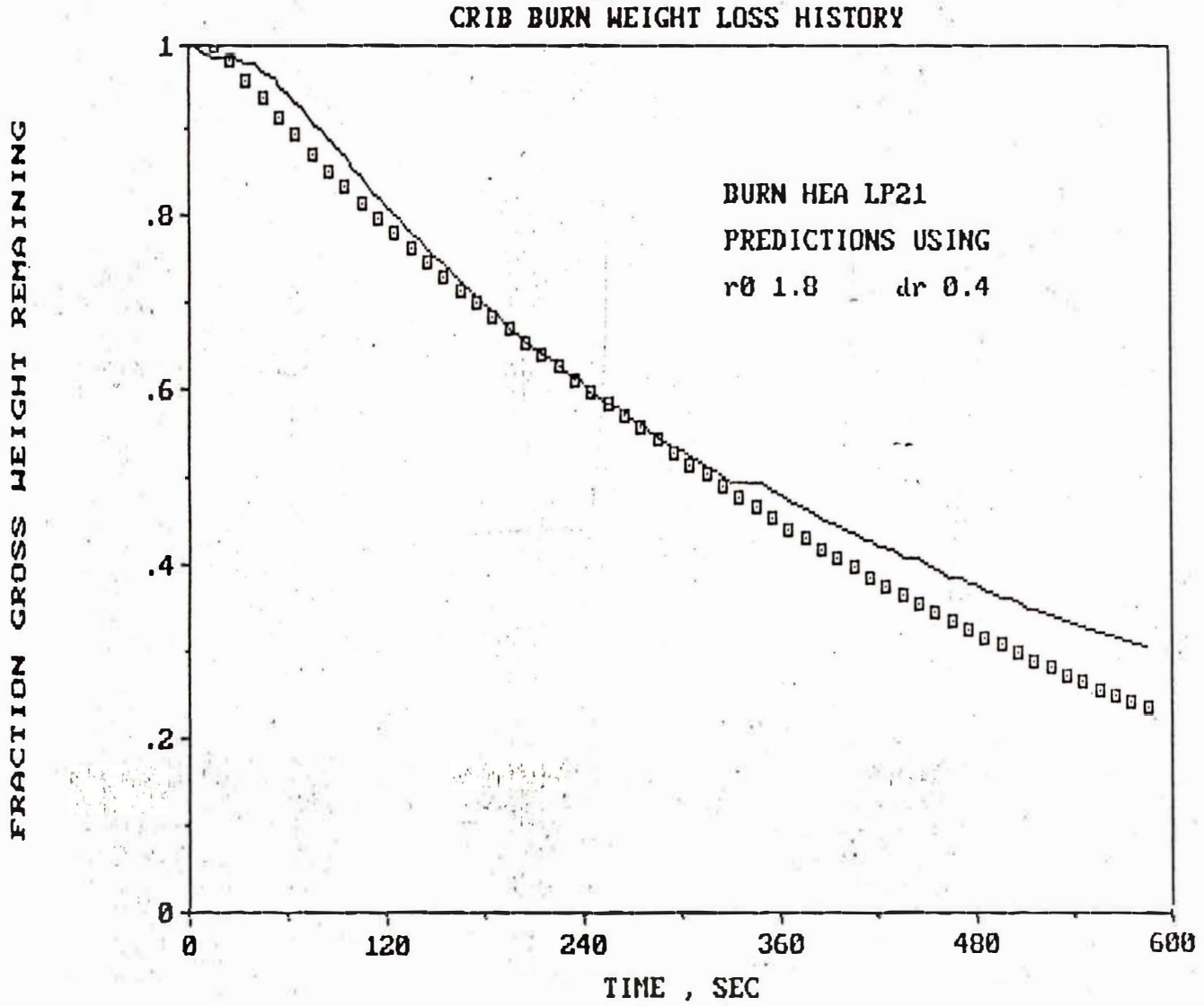

CRIB BURN WEIGHT LOSS HISTORY
FRACTION GROSS WEIGHT REMAINING
1
.8
.6
.4
.2
0
0
120
240
360
480
600
TIME , SEC
BURN HEA 2T111
PREDICTIONS USING
r0 1.8
dr 0.4

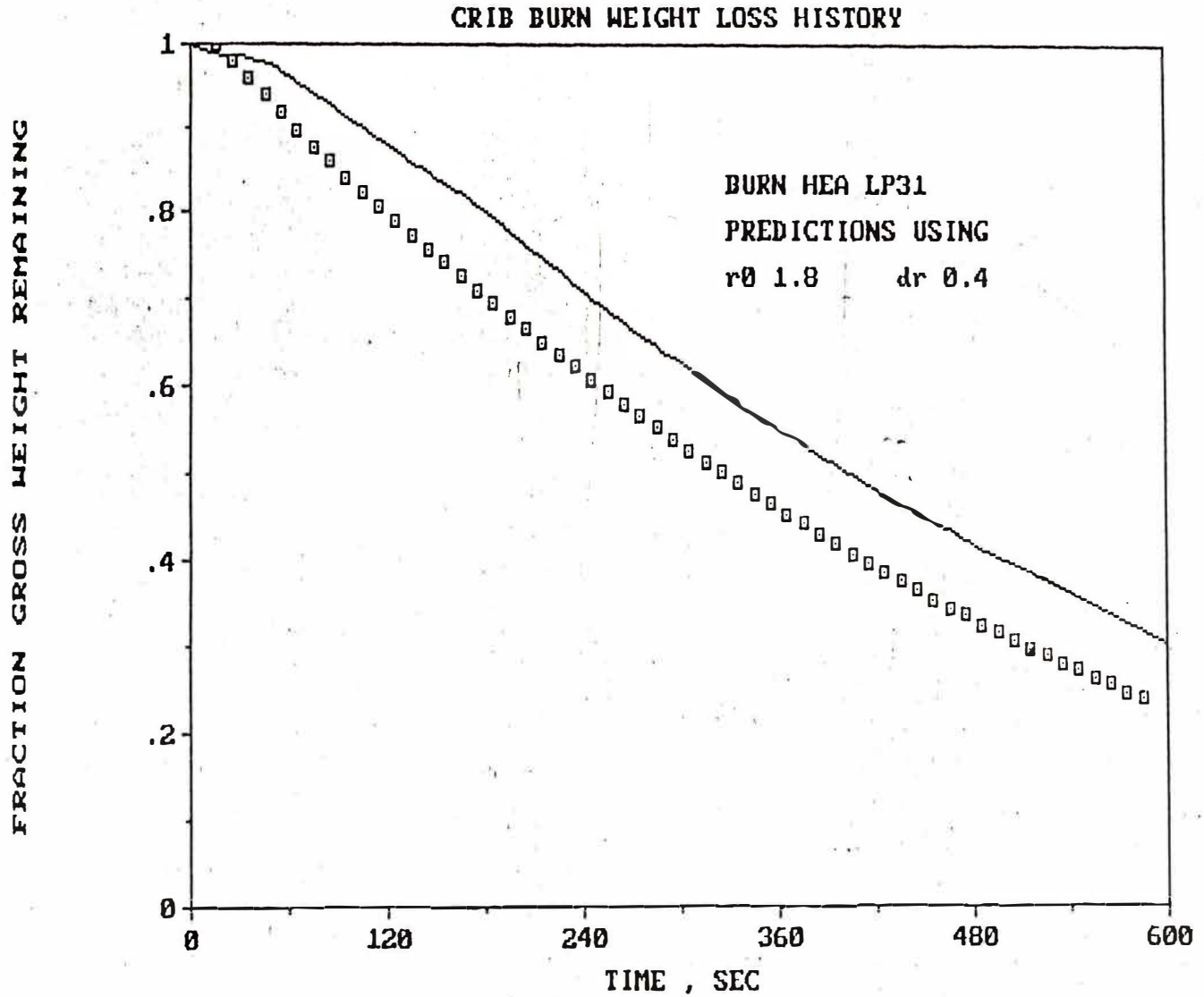

CRIB BURN WEIGHT LOSS HISTORY
FRACTION GROSS WEIGHT REMAINING
1
.8
.6
.4
.2
0
0
120
240
360
480
600
TIME , SEC
BURN HEA 3T131
PREDICTIONS USING
r0 1.75 dr 0.40

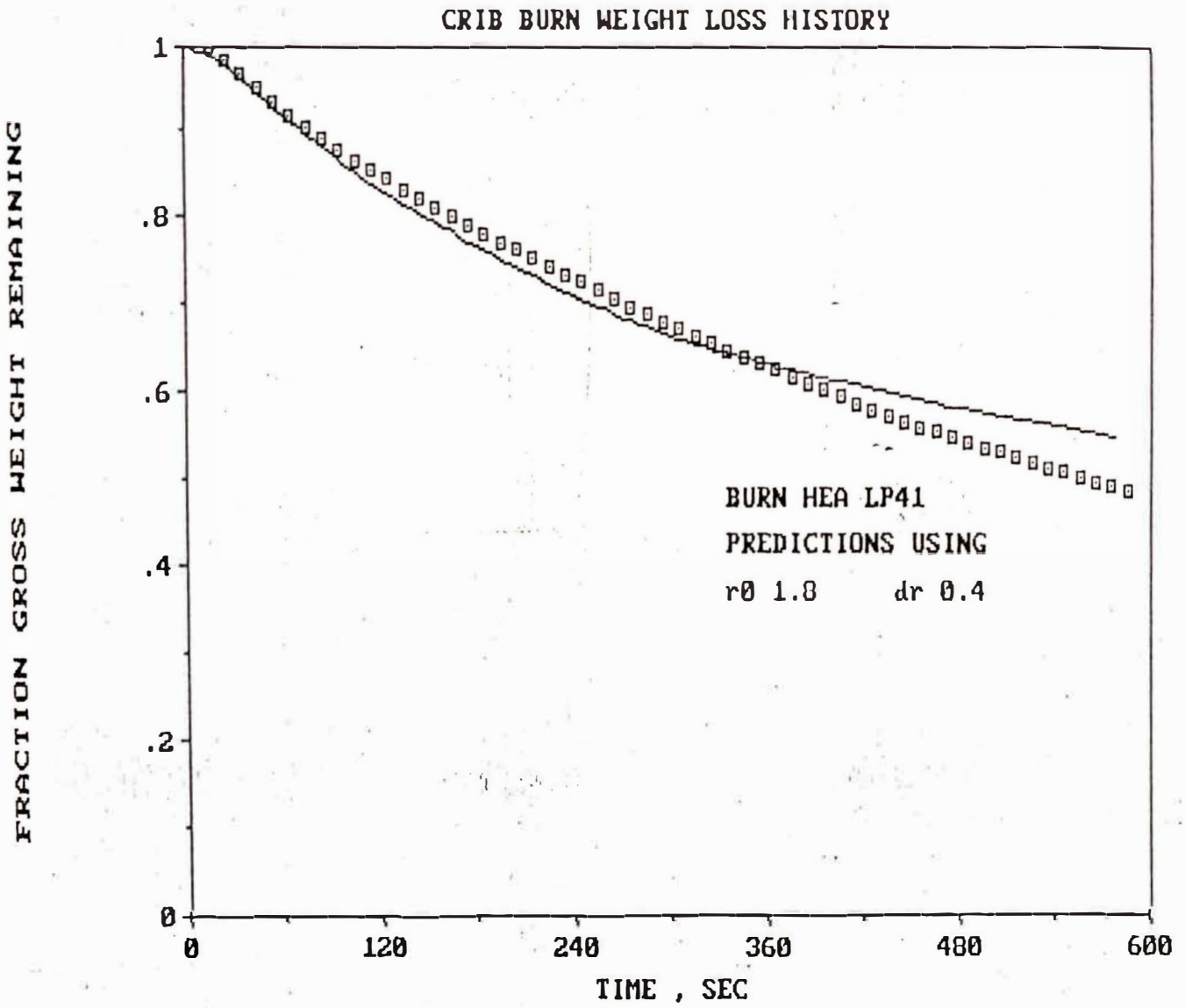

CRIB BURN WEIGHT LOSS HISTORY
FRACTION GROSS WEIGHT REMAINING
1
.8
.6
.4
.2
0
0
120
240
360
480
600
TIME , SEC
BURN HEA 3T131
PREDICTIONS USING
r0 1.8 dr 0.4

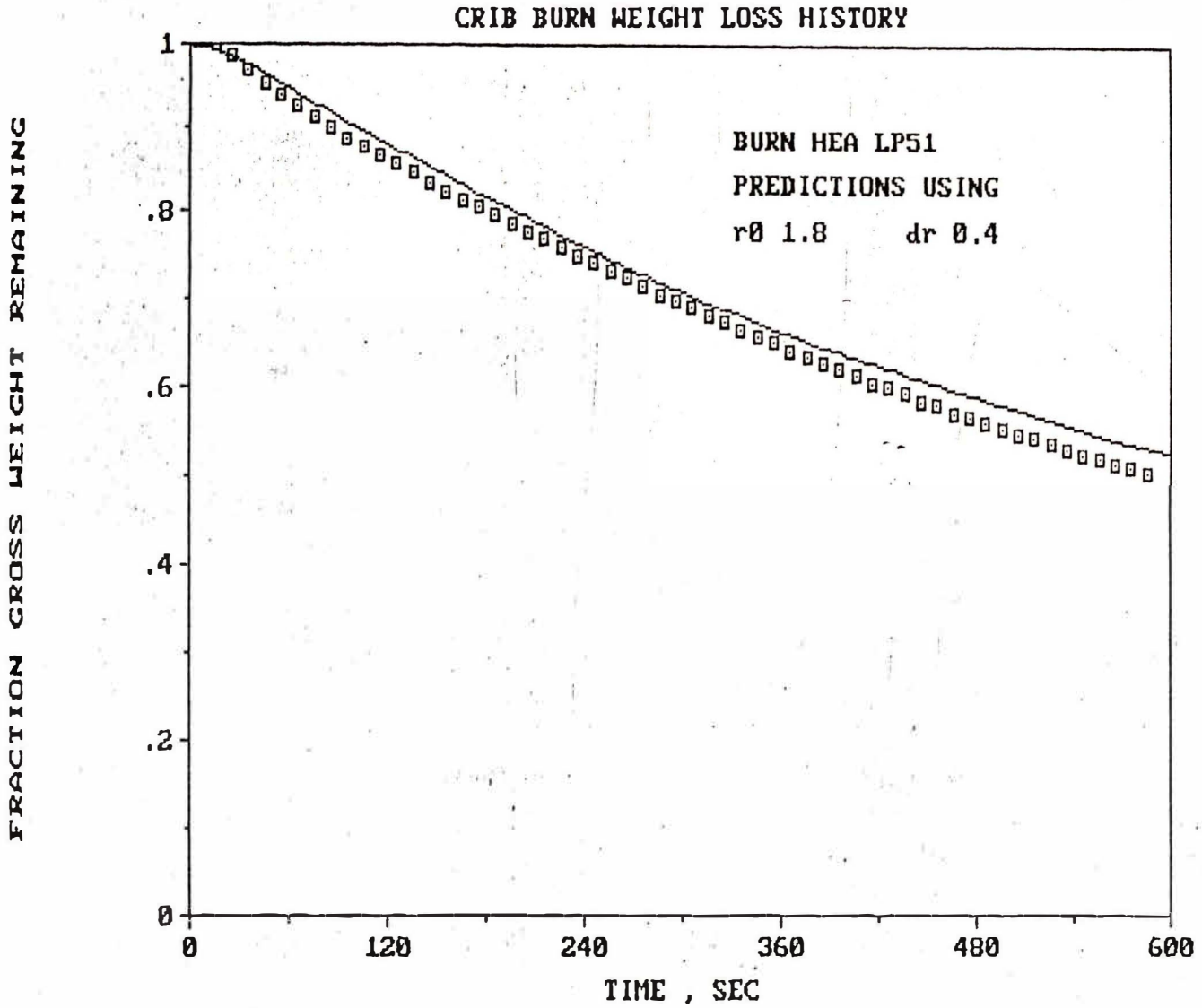

CRIB BURN WEIGHT LOSS HISTORY
FRACTION GROSS WEIGHT REMAINING
1
.8
.6
.4
.2
0
BURN HEA DF21
PREDICTIONS USING
r0 1.8 dr 0.4
0
120
240
360
480
600
TIME , SEC

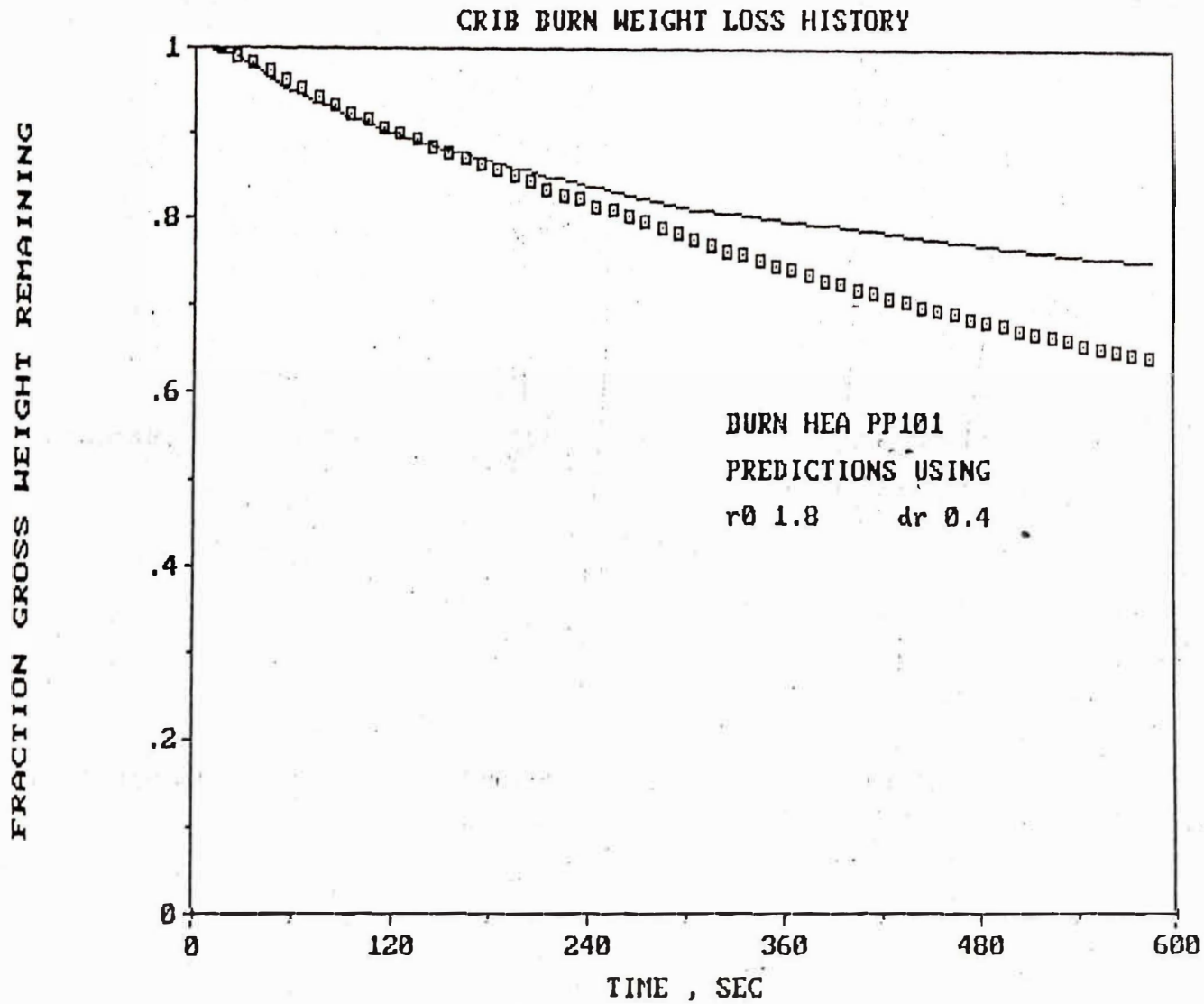

CRIB BURN WEIGHT LOSS HISTORY
FRACTION GROSS WEIGHT REMAINING
1
.8
.6
.4
.2
0
0
120
240
360
480
600
TIME , SEC
BURN HEA DF41
PREDICTIONS USING
r0 1.8
dr 0.4

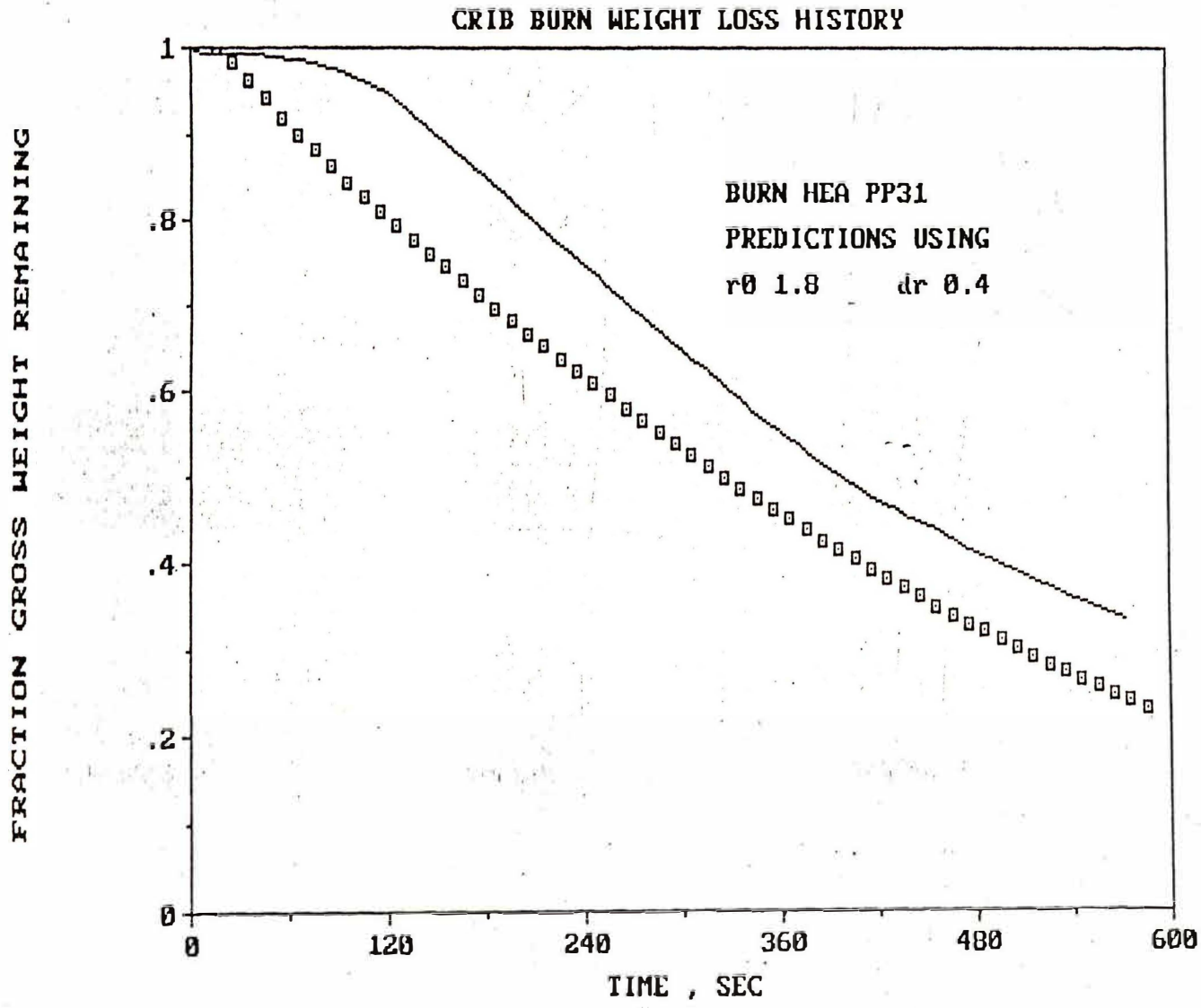

CRIB BURN WEIGHT LOSS HISTORY
FRACTION GROSS WEIGHT REMAINING
1
.8
.6
.4
.2
0
0
120
240
360
480
600
TIME , SEC
BURN HEA LP21
PREDICTIONS USING
r0 1.8
dr 0.4

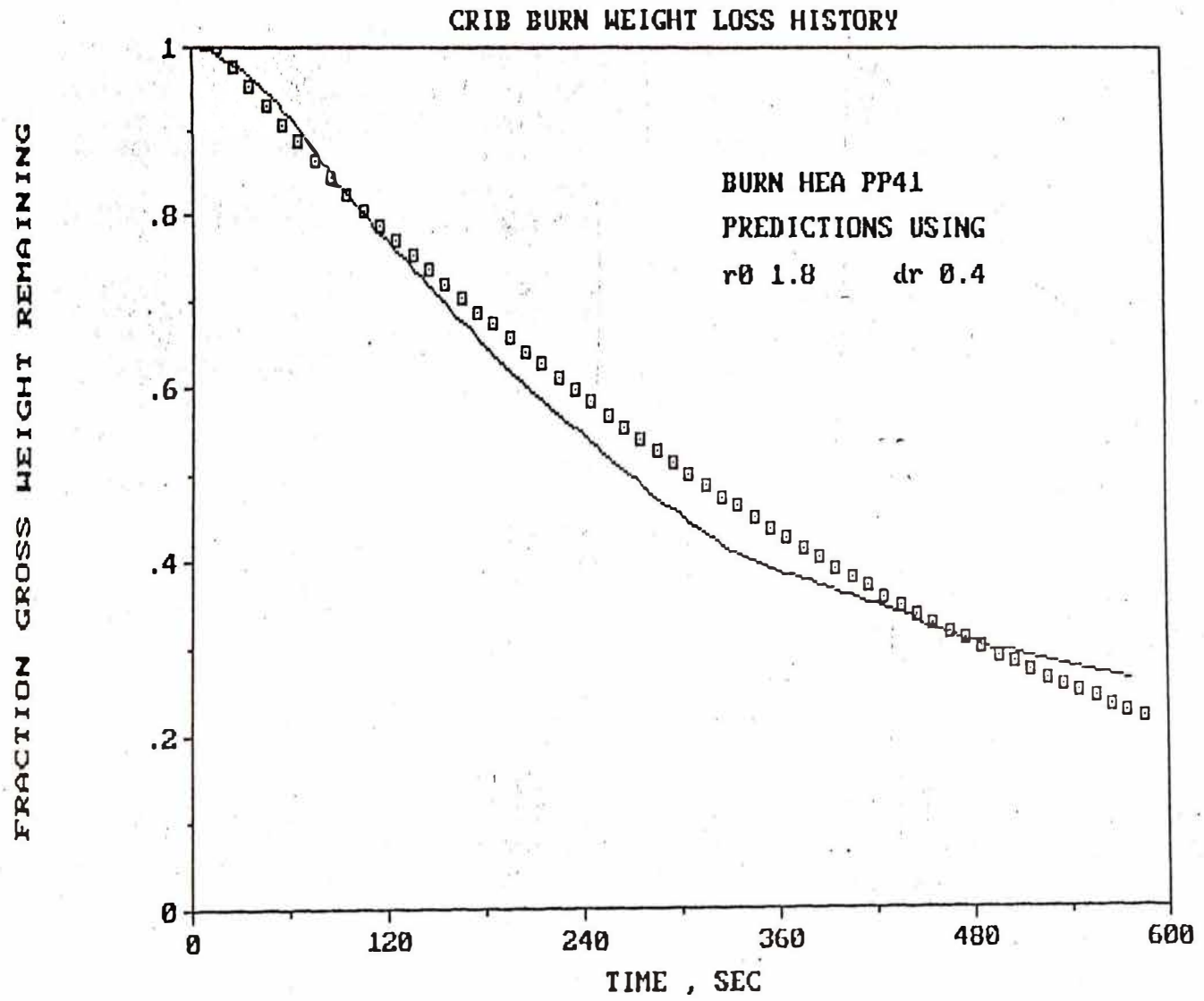

CRIB BURN WEIGHT LOSS HISTORY
FRACTION GROSS WEIGHT REMAINING
1
.8
.6
.4
.2
0
0
120
240
360
480
600
TIME , SEC
BURN HEA LP31
PREDICTIONS USING
r0 1.8
dr 0.4

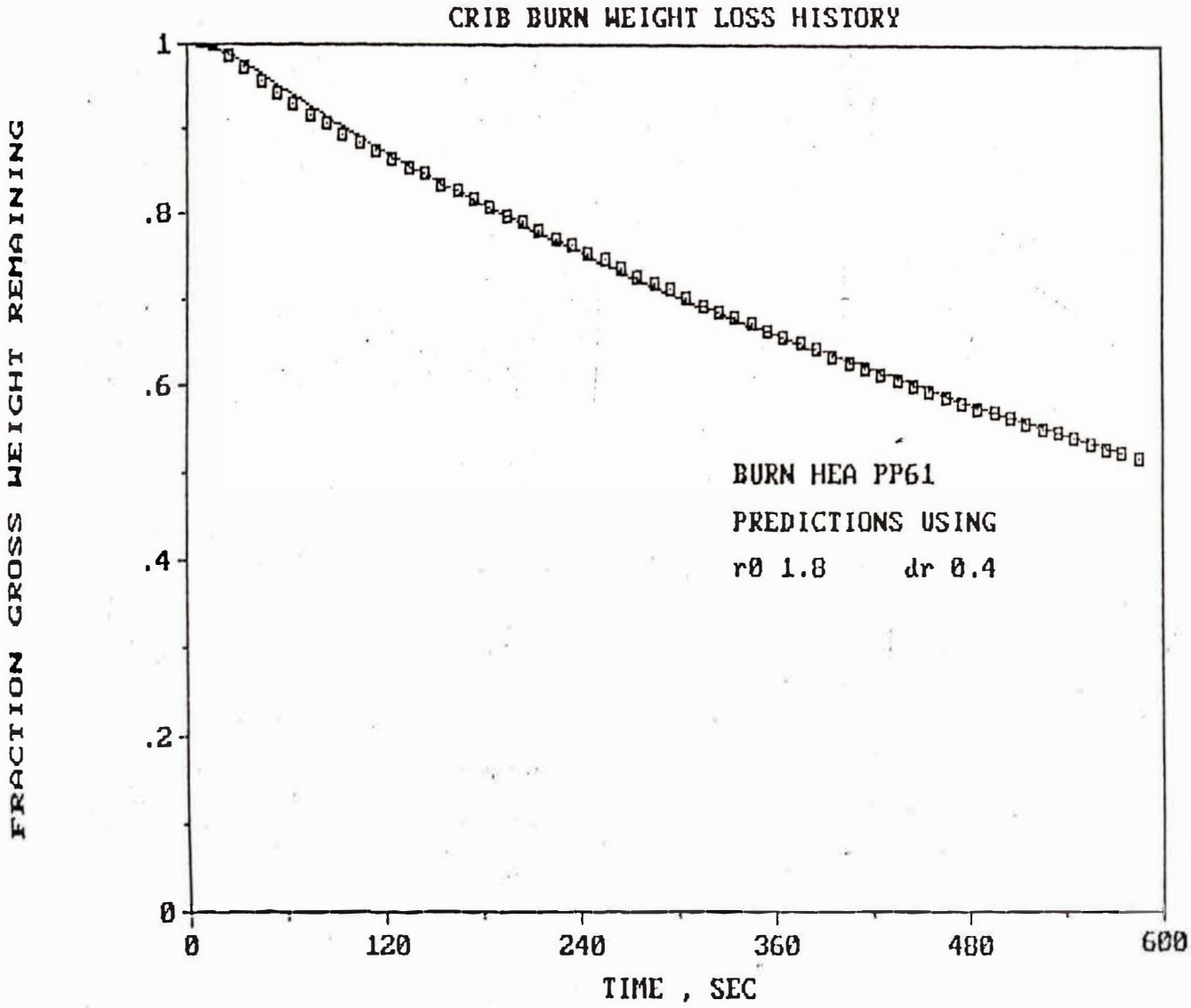

CRIB BURN WEIGHT LOSS HISTORY
FRACTION GROSS WEIGHT REMAINING
1
.8
.6
.4
.2
0
0
120
240
360
480
600
TIME , SEC
BURN HEA LP41
PREDICTIONS USING
r0 1.8
dr 0.4

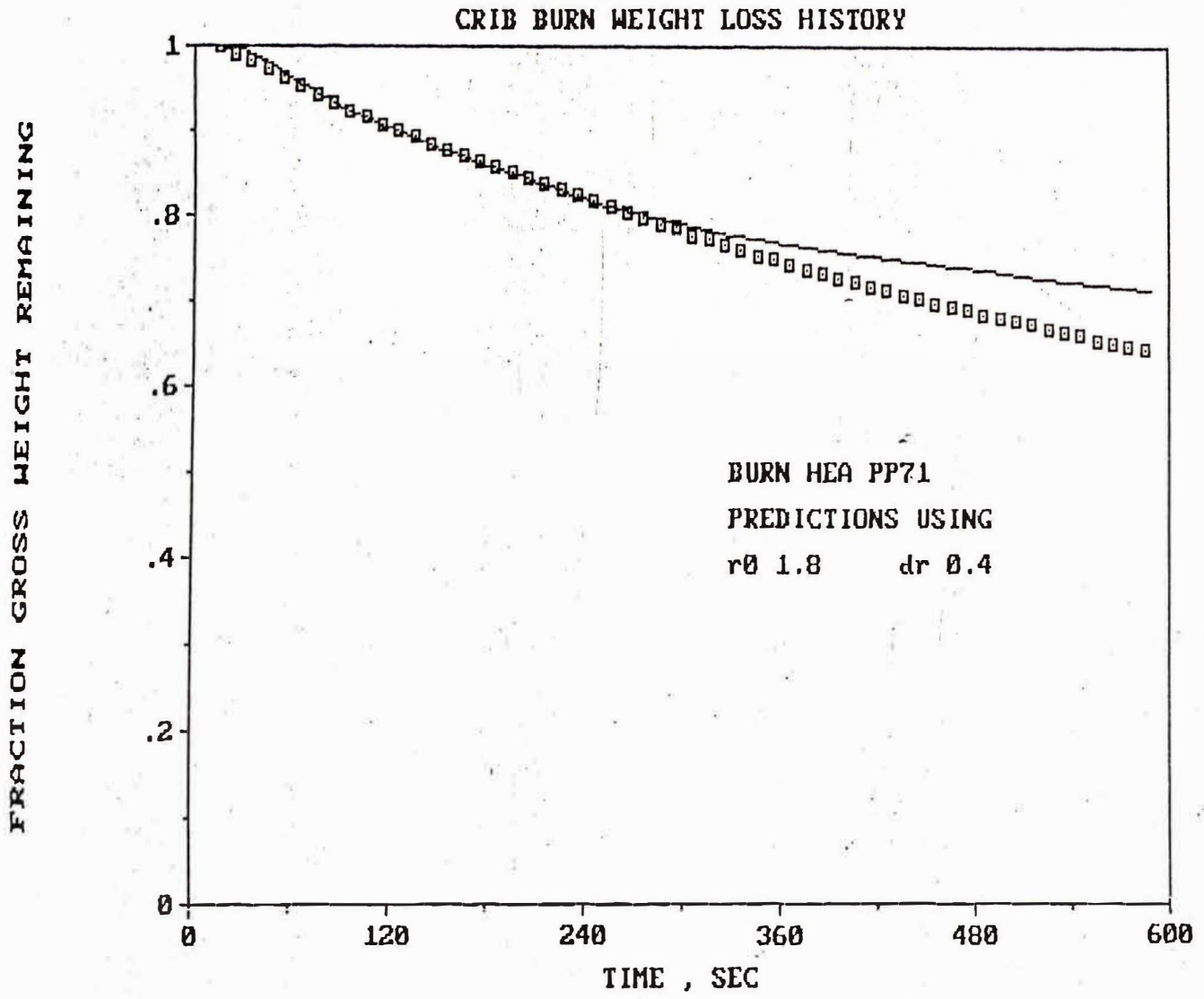

CRIB BURN WEIGHT LOSS HISTORY
FRACTION GROSS WEIGHT REMAINING
1
.8
.6
.4
.2
0
BURN HEA LP51
PREDICTIONS USING
r0 1.8 dr 0.4
0
120
240
360
480
600
TIME , SEC

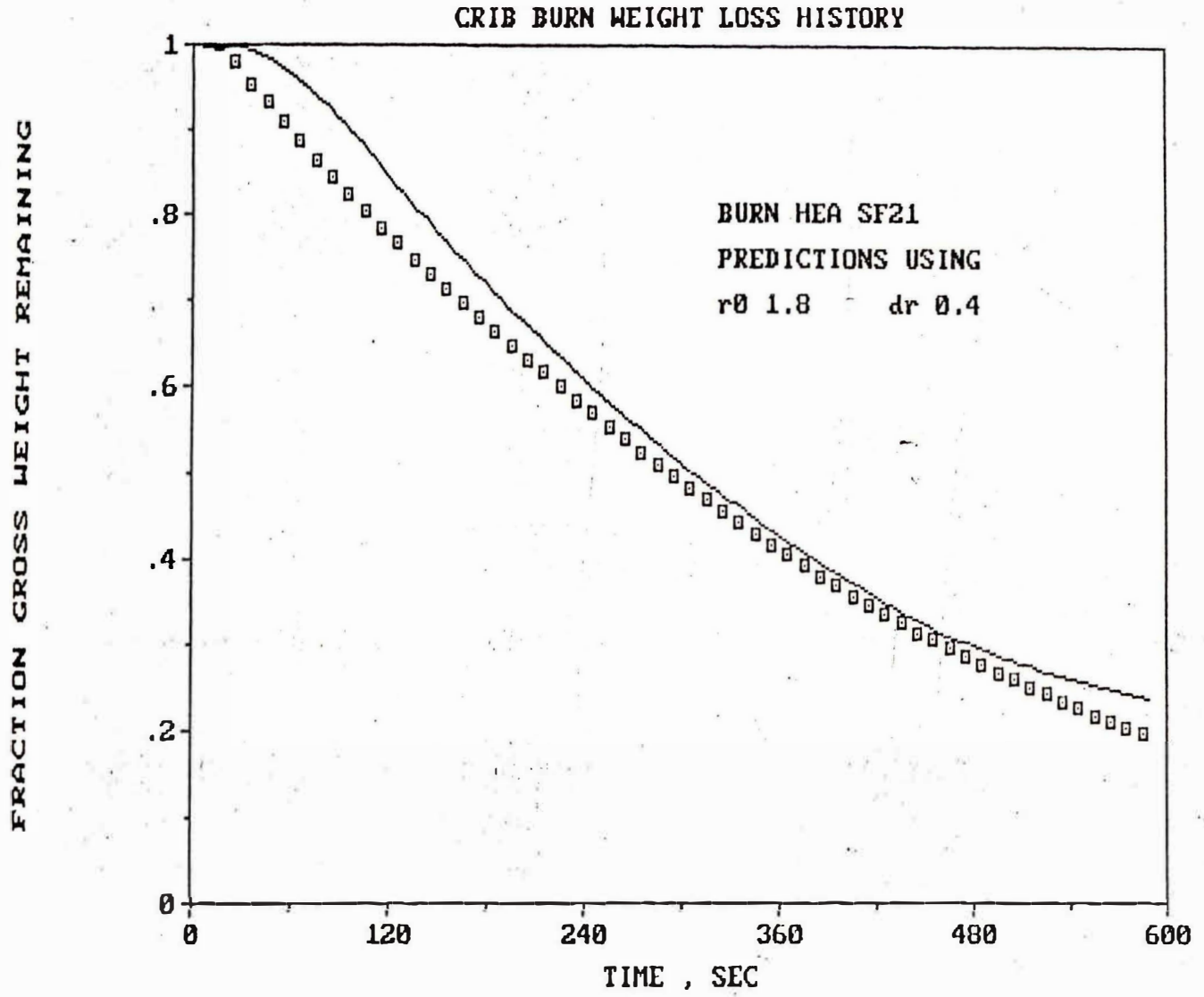

CRIB BURN WEIGHT LOSS HISTORY
FRACTION GROSS WEIGHT REMAINING
1
.8
.6
.4
.2
0
0
120
240
360
480
600
TIME , SEC
BURN HEA PP101
PREDICTIONS USING
r0 1.8 dr 0.4

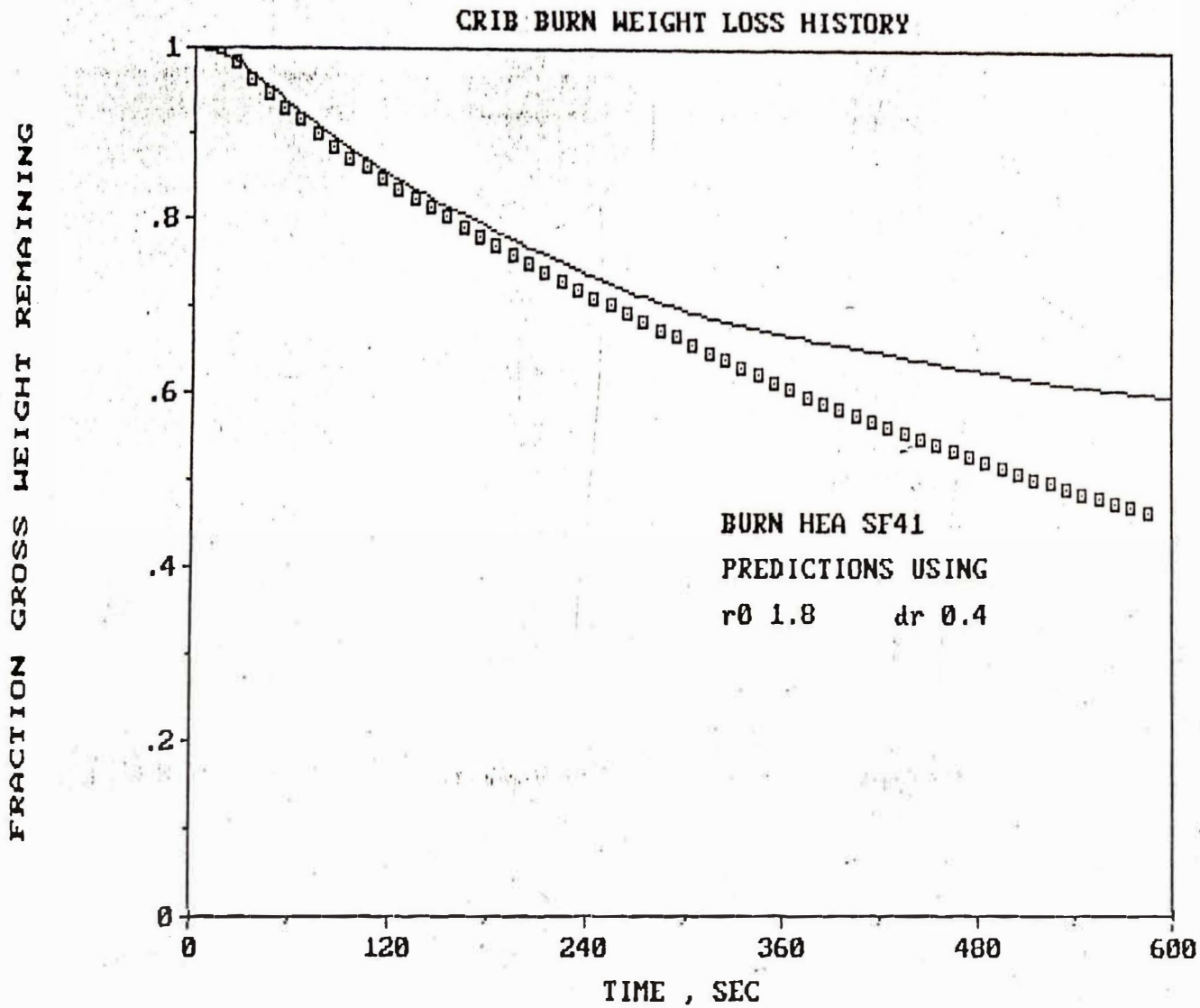

CRIB BURN WEIGHT LOSS HISTORY
FRACTION GROSS WEIGHT REMAINING
1
.8
.6
.4
.2
0
0
120
240
360
480
600
TIME , SEC
BURN HEA PP31
PREDICTIONS USING
r0 1.8
dr 0.4


CRIB BURN WEIGHT LOSS HISTORY
FRACTION GROSS WEIGHT REMAINING
1
.8
.6
.4
.2
0
0
120
240
360
480
600
TIME , SEC
BURN HEA PP41
PREDICTIONS USING
r0 1.8
dr 0.4


CRIB BURN WEIGHT LOSS HISTORY
FRACTION GROSS WEIGHT REMAINING
1
.8
.6
.4
.2
0
0
120
240
360
480
600
TIME , SEC
BURN HEA PP61
PREDICTIONS USING
r0 1.8 dr 0.4


CRIB BURN WEIGHT LOSS HISTORY
FRACTION GROSS WEIGHT REMAINING
1
.8
.6
.4
.2
0
0
120
240
360
480
600
TIME , SEC
BURN HEA PP71
PREDICTIONS USING
r0 1.8
dr 0.4


CRIB BURN WEIGHT LOSS HISTORY
FRACTION GROSS WEIGHT REMAINING
1
.8
.6
.4
.2
0
0
120
240
360
480
600
TIME , SEC
BURN HEA SF21
PREDICTIONS USING
r0 1.8
dr 0.4


CRIB BURN WEIGHT LOSS HISTORY
FRACTION GROSS WEIGHT REMAINING
1
.8
.6
.4
.2
0
0
120
240
360
480
600
TIME , SEC
BURN HEA SF41
PREDICTIONS USING
r0 1.8
dr 0.4

CRIB BURNS WITH RATE CONTROLLED BY TOP SURFACE VENTILLATION AREA

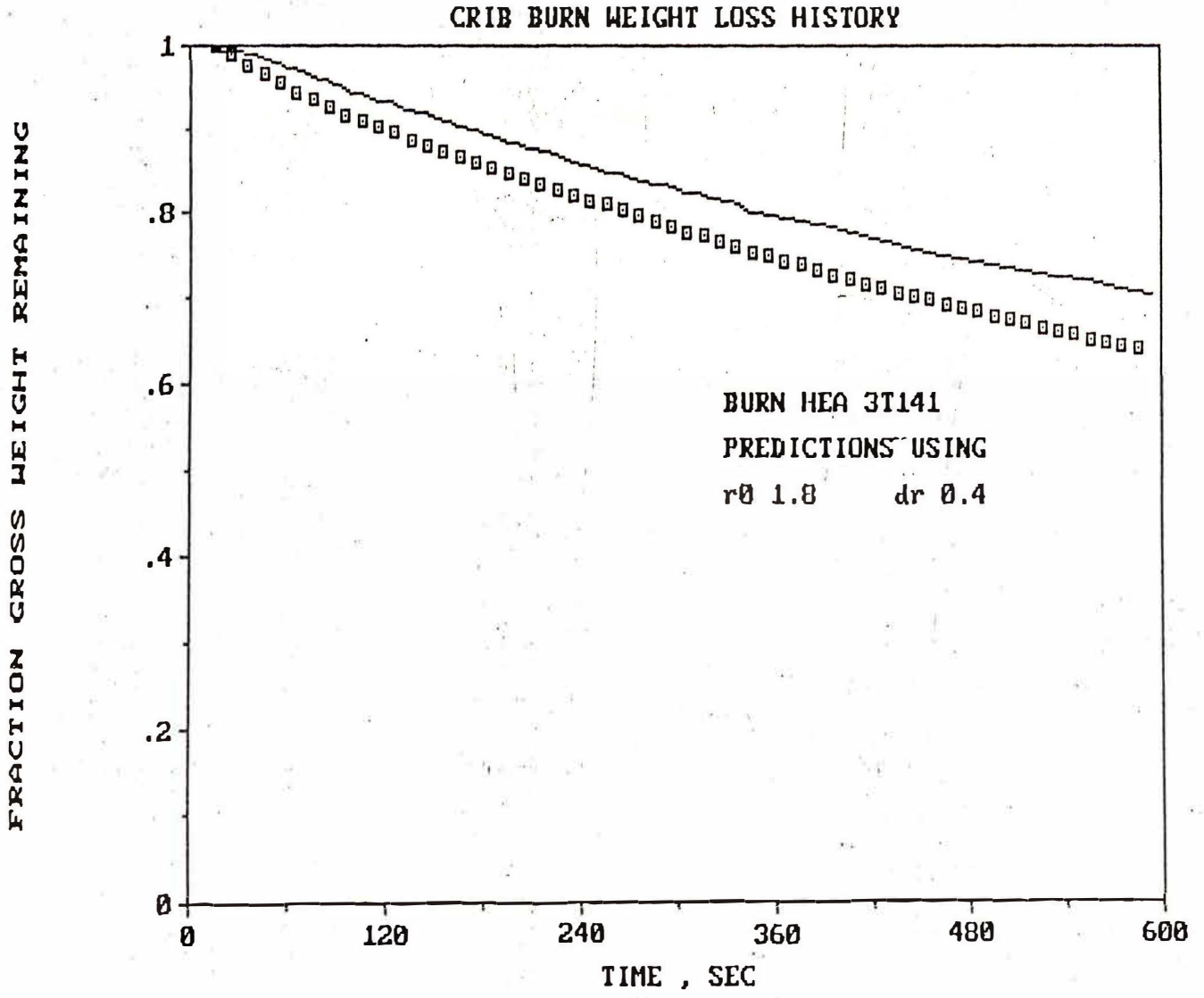
BURN ID	Dry Weight, kg	Principal fuel
HEA3T141	235.0	Ponderosa pine
HEALP61	95.9	Lodgepole pine
HEAPP81	163.7	Ponderosa pine
HEAPP91	217.7	Ponderosa pine
HEASF31	82.0	Spruce / fir
HEASF51	161.0	Spruce / fir

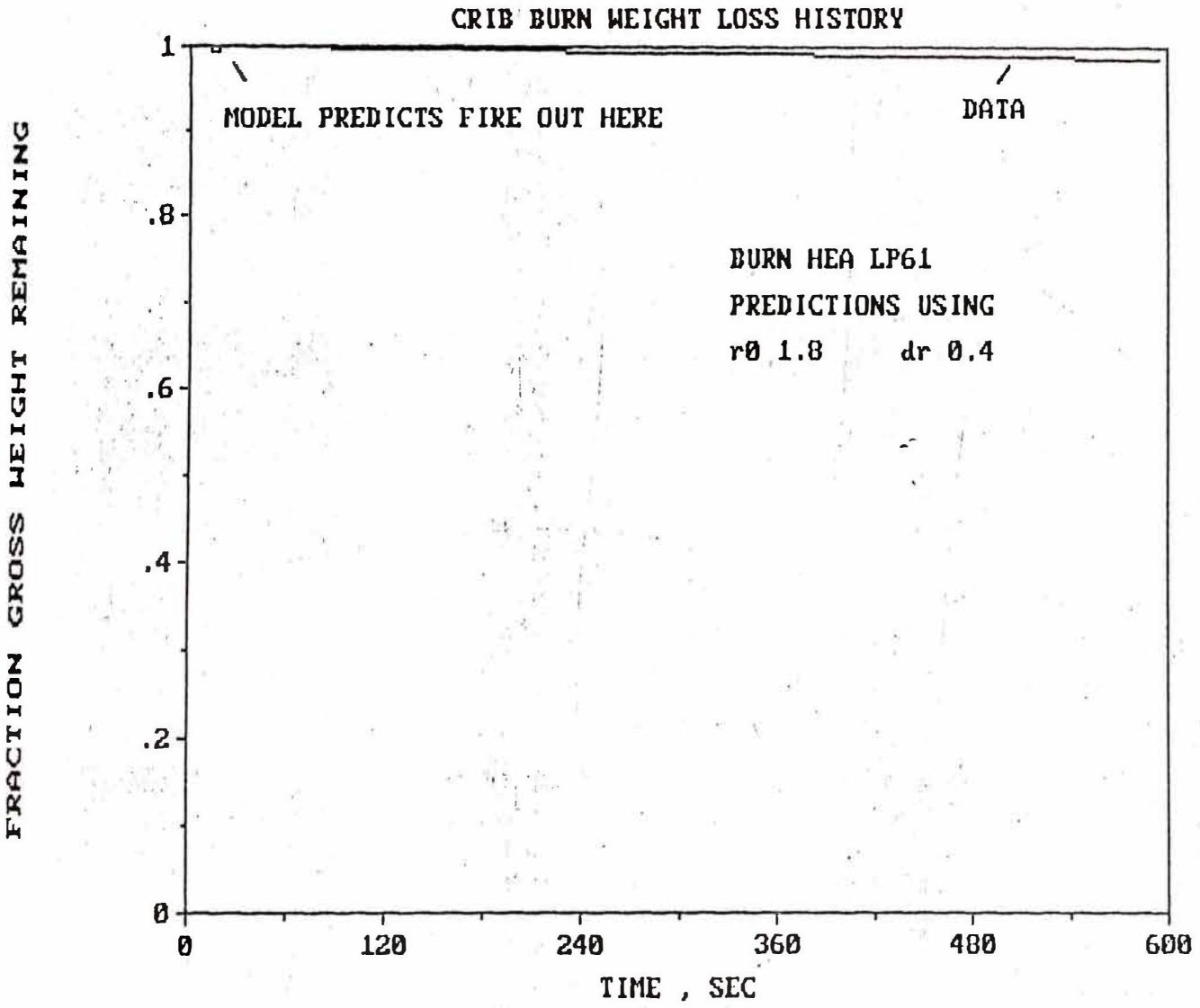
CRIB BURNS WITH RATE CONTROL MECHANISM UNCERTAIN (MARGINAL CASES)

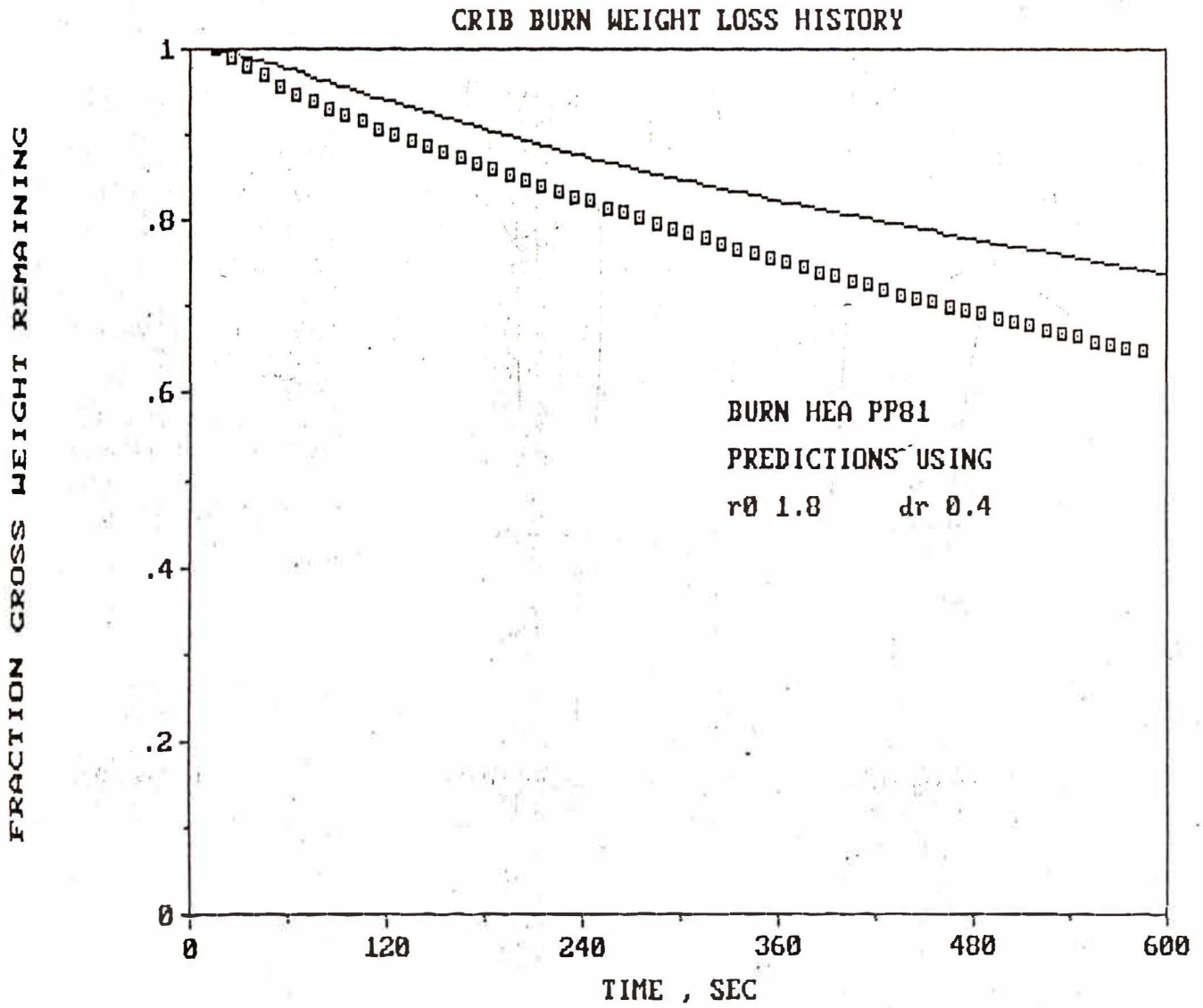
HEA3T151	188.7	Ponderosa pine
HEA3T161	81.2	Ponderosa pine
HEADF31	81.1	Douglas fir
HEADF51	152.4	Douglas fir
HEADF52	152.4	Douglas fir
HEAPP21	46.3	Ponderosa pine

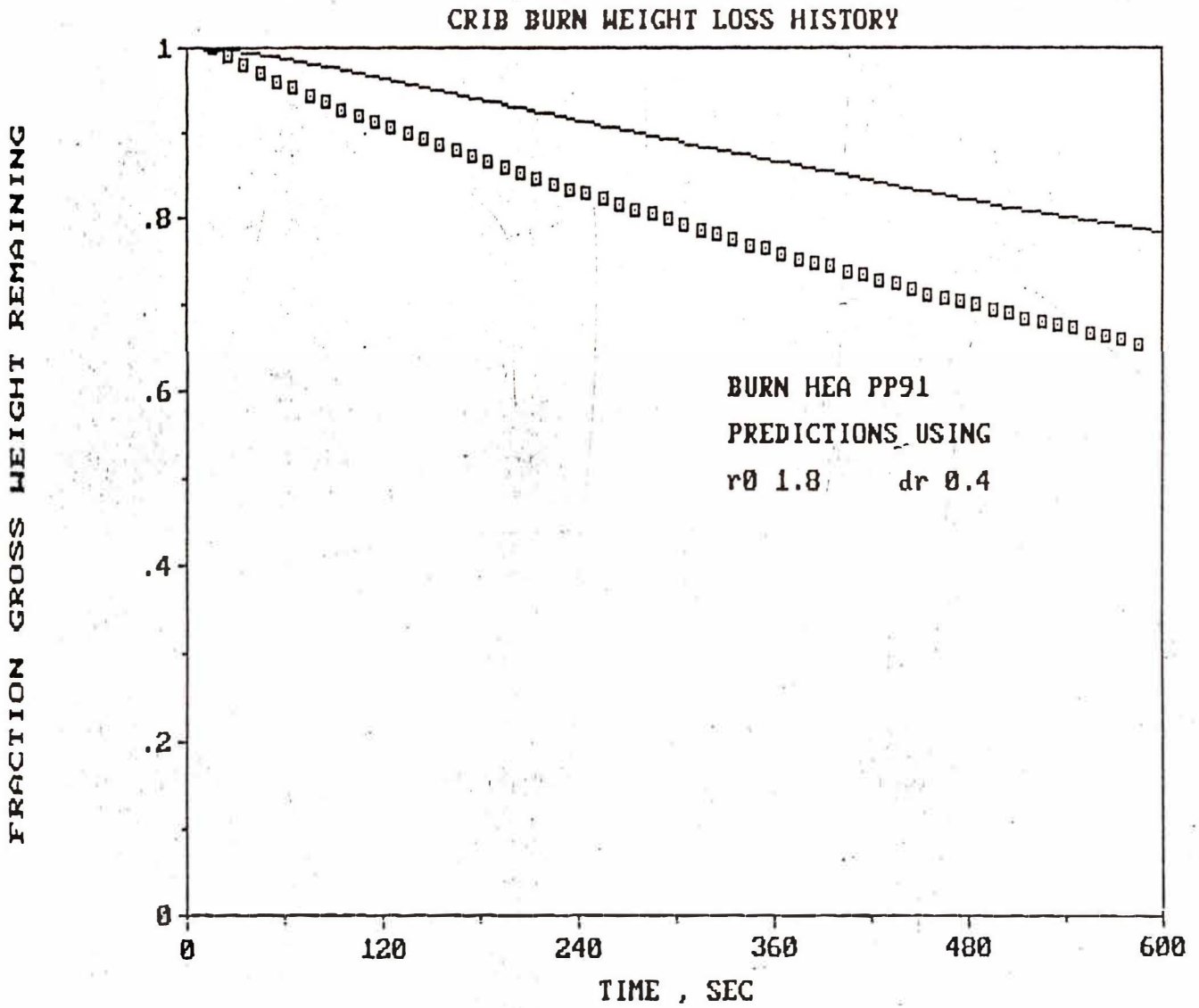
CRIB BURNS WITH RATE CONTROL MECHANISM UNDETERMINED (PROBABLY SA)

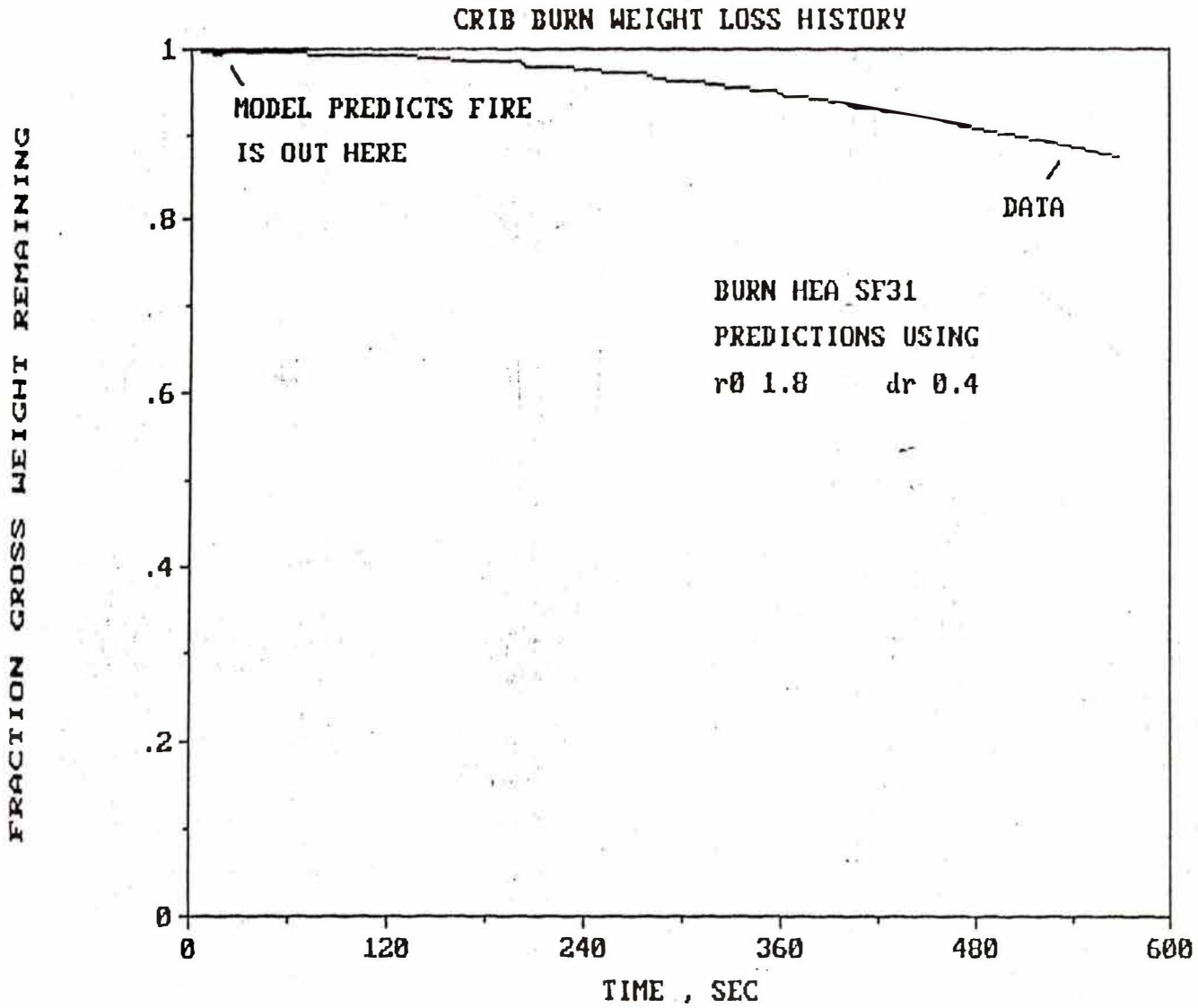
HEA1T014	3.5	Ponderosa pine
HEA1T015	25.5	Ponderosa pine
HEA3T171	23.9	Ponderosa pine
HEA3T172	28.3	Ponderosa pine
HEA3T173	25.5	Ponderosa pine

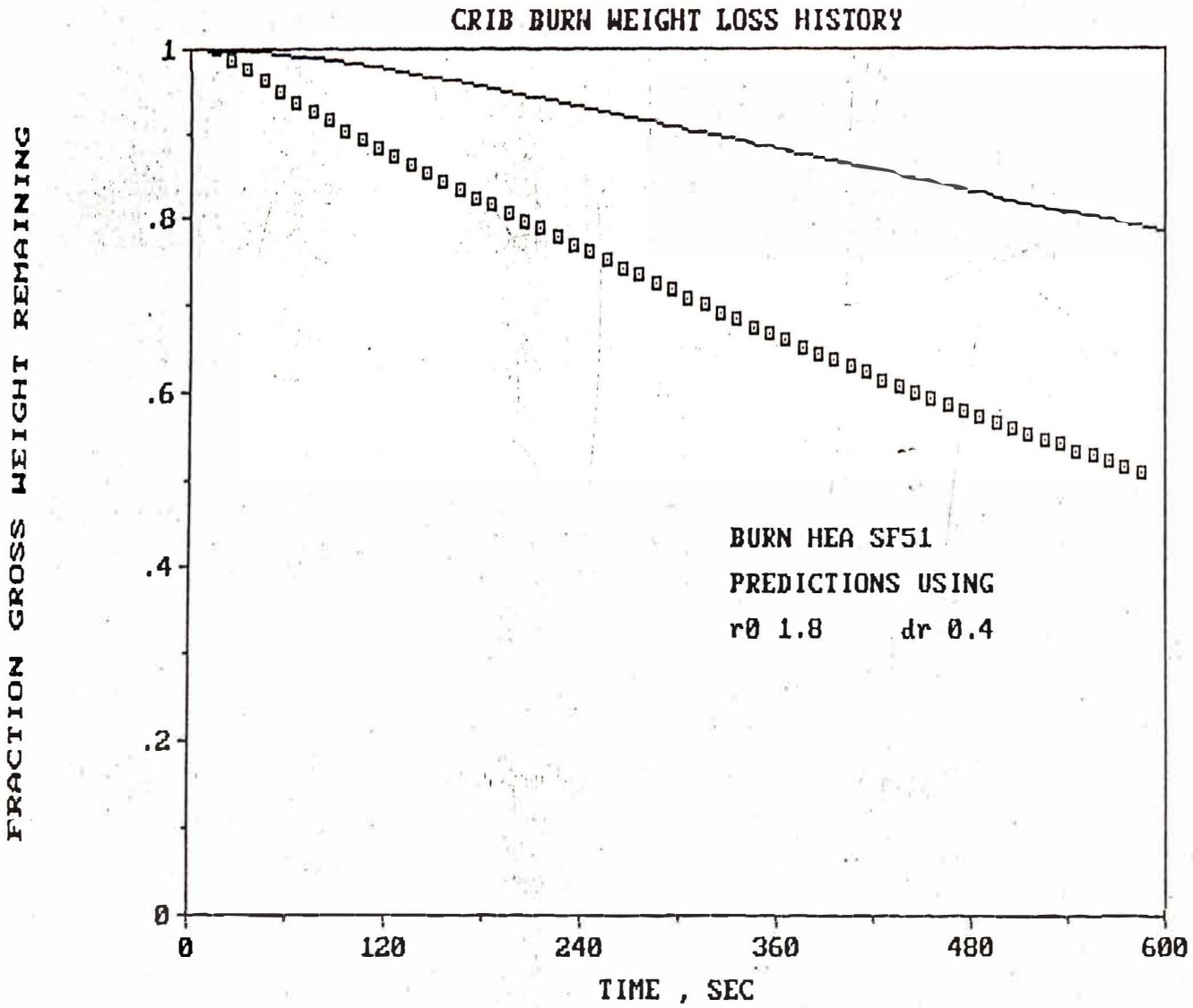

CRIB BURN WEIGHT LOSS HISTORY
FRACTION GROSS WEIGHT REMAINING
1
.8
.6
.4
.2
0
0
120
240
360
480
600
TIME , SEC
BURN HEA 3T141
PREDICTIONS USING
r0 1.8 dr 0.4

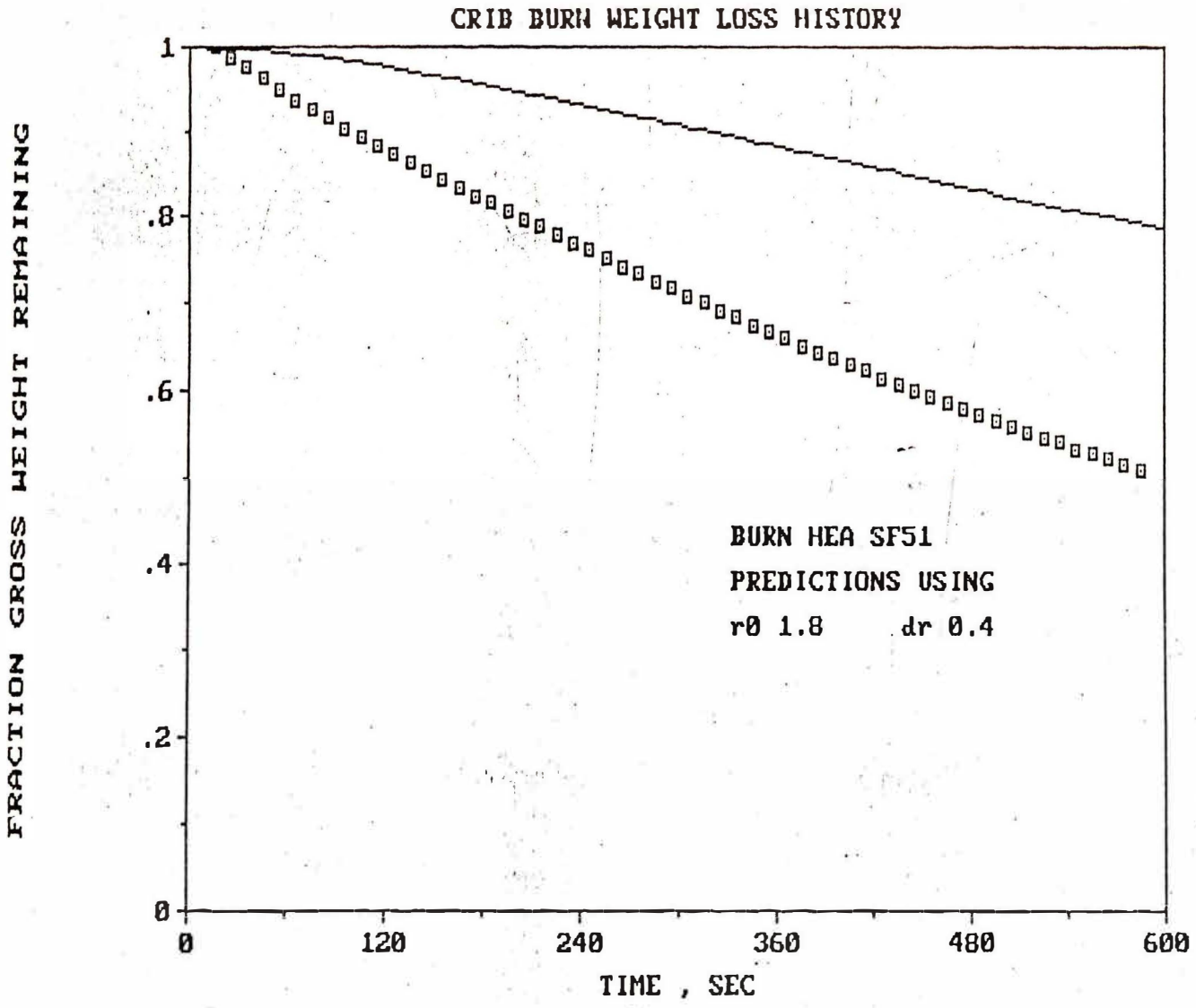

CRIB BURN WEIGHT LOSS HISTORY
MODEL PREDICTS FIRE OUT HERE
DATA
BURN HEA LP61
PREDICTIONS USING
r0 1.8 dr 0.4
FRACTION GROSS WEIGHT REMAINING
1
.8
.6
.4
.2
0
0
120
240
360
480
600
TIME , SEC

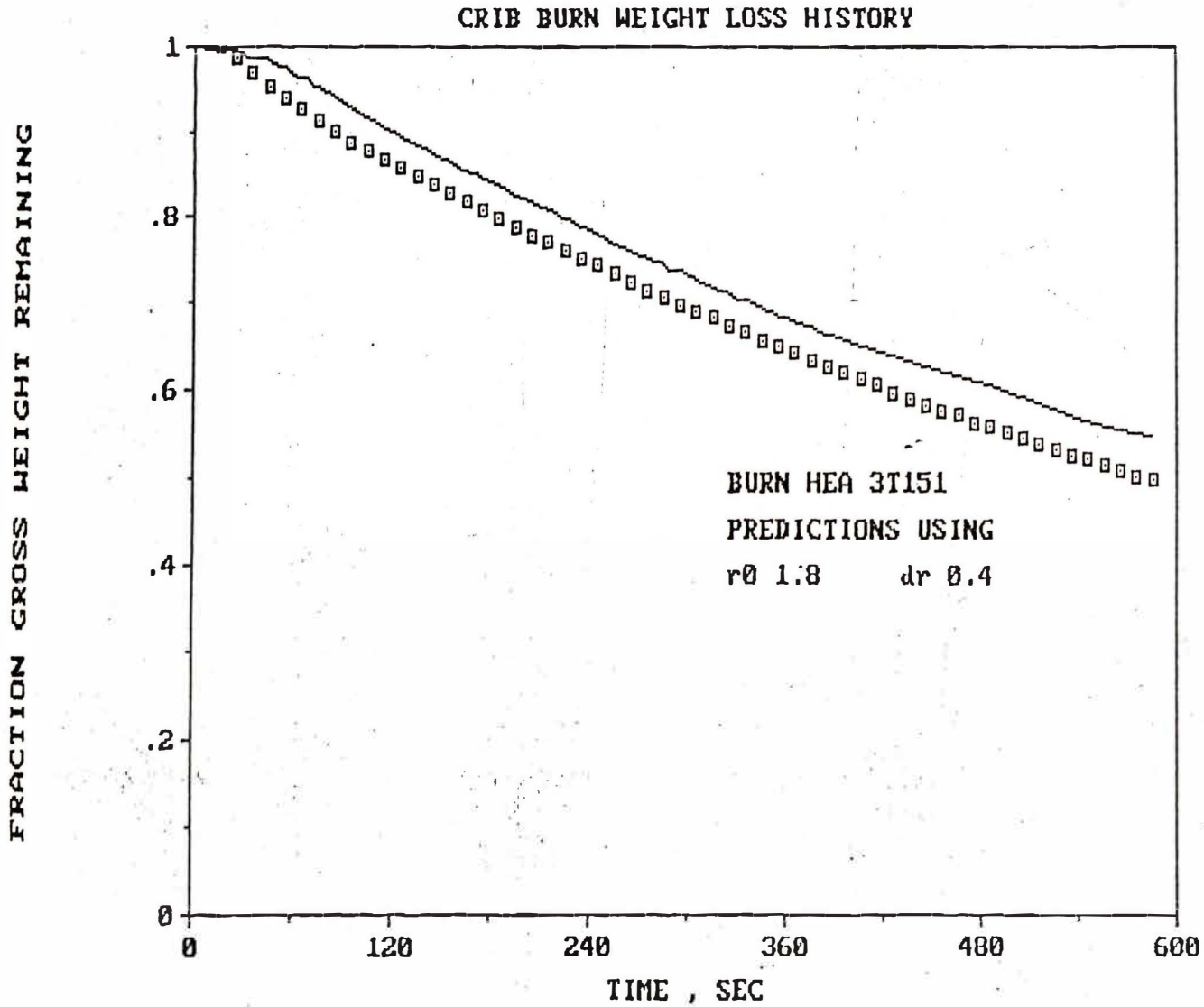

CRIB BURN WEIGHT LOSS HISTORY
FRACTION GROSS WEIGHT REMAINING
1
.8
.6
.4
.2
0
0
120
240
360
480
600
TIME , SEC
BURN HEA PP81
PREDICTIONS USING
r0 1.8 dr 0.4

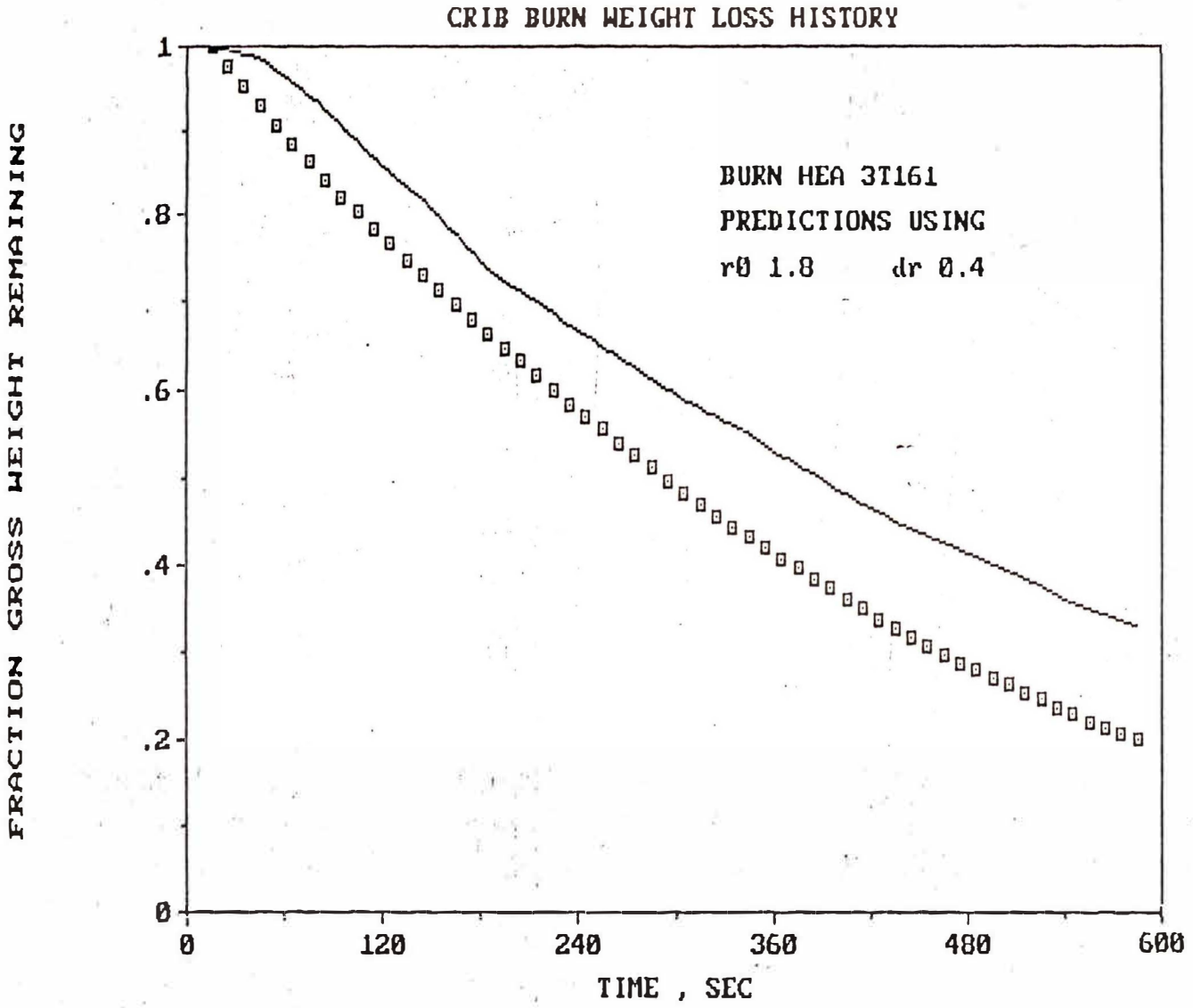

CRIB BURN WEIGHT LOSS HISTORY
FRACTION GROSS WEIGHT REMAINING
1
.8
.6
.4
.2
0
0
120
240
360
480
600
TIME , SEC
BURN HEA PP91
PREDICTIONS USING
r0 1.8
dr 0.4

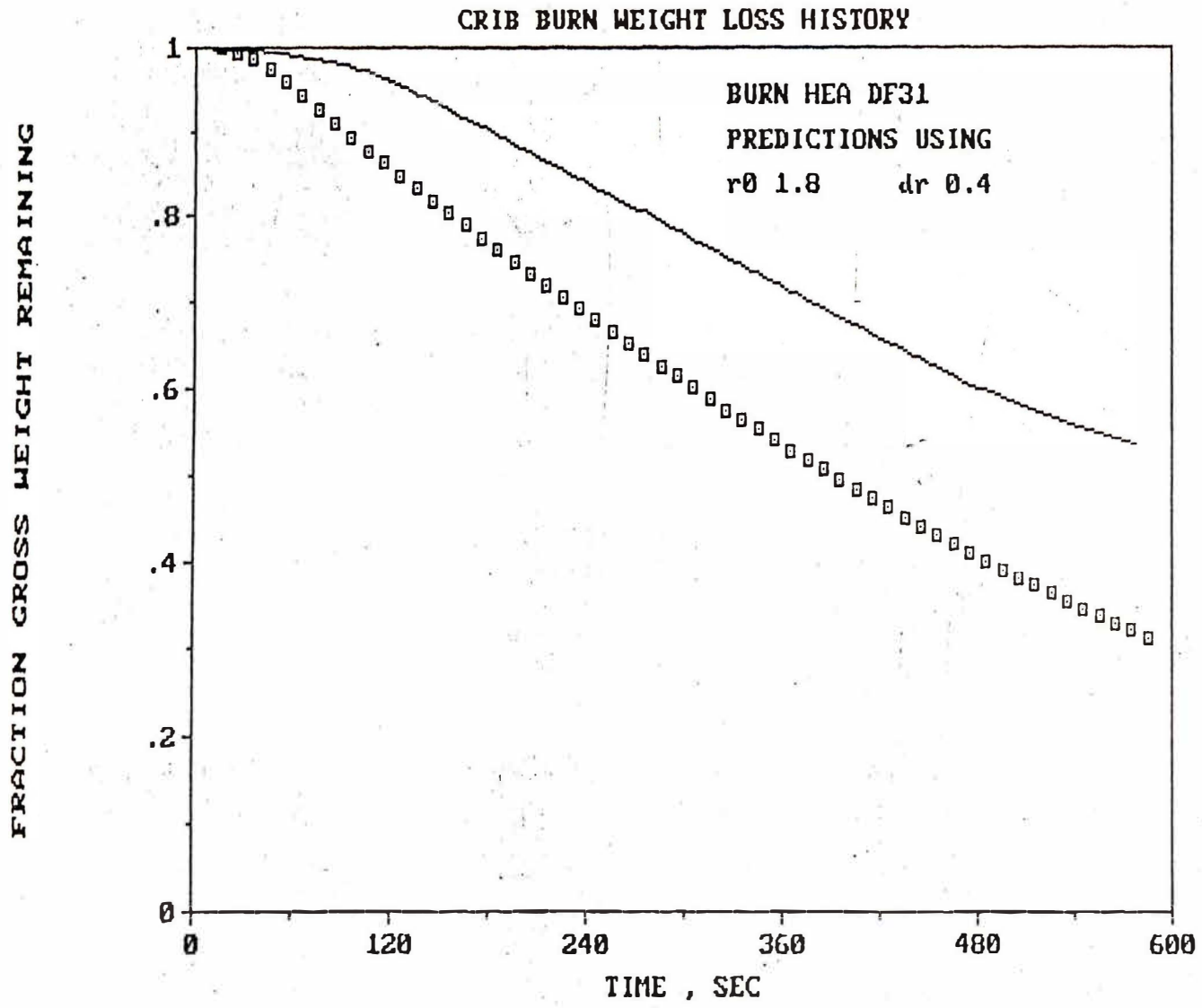

CRIB BURN WEIGHT LOSS HISTORY
FRACTION GROSS WEIGHT REMAINING
1
.8
.6
.4
.2
0
MODEL PREDICTS FIRE
IS OUT HERE
DATA
BURN HEA SF31
PREDICTIONS USING
r0 1.8 dr 0.4
0
120
240
360
480
600
TIME , SEC

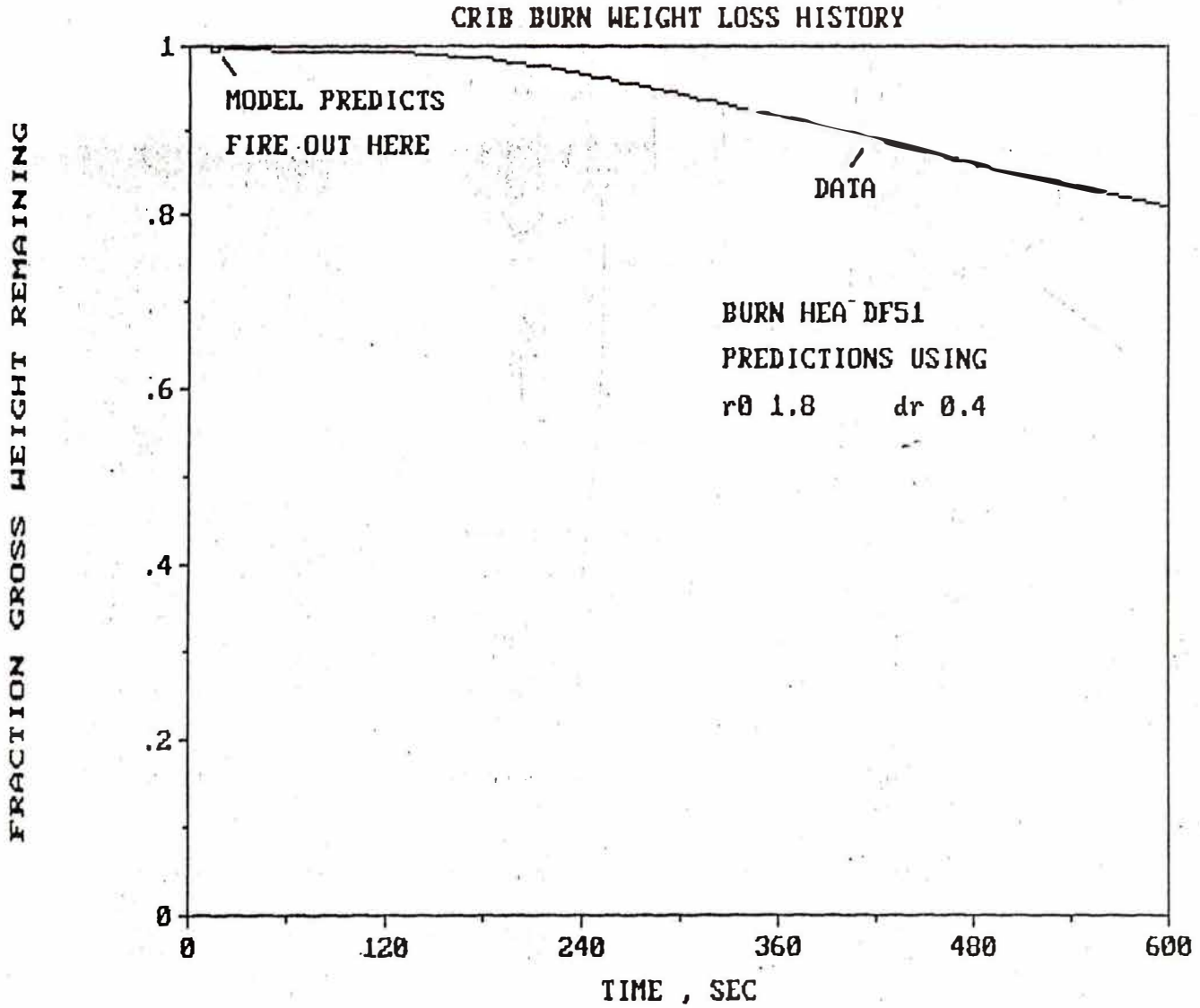

CRIB BURN WEIGHT LOSS HISTORY
FRACTION GROSS WEIGHT REMAINING
1
.8
.6
.4
.2
0
0
120
240
360
480
600
TIME , SEC
BURN HEA SF51
PREDICTIONS USING
r0 1.8 dr 0.4

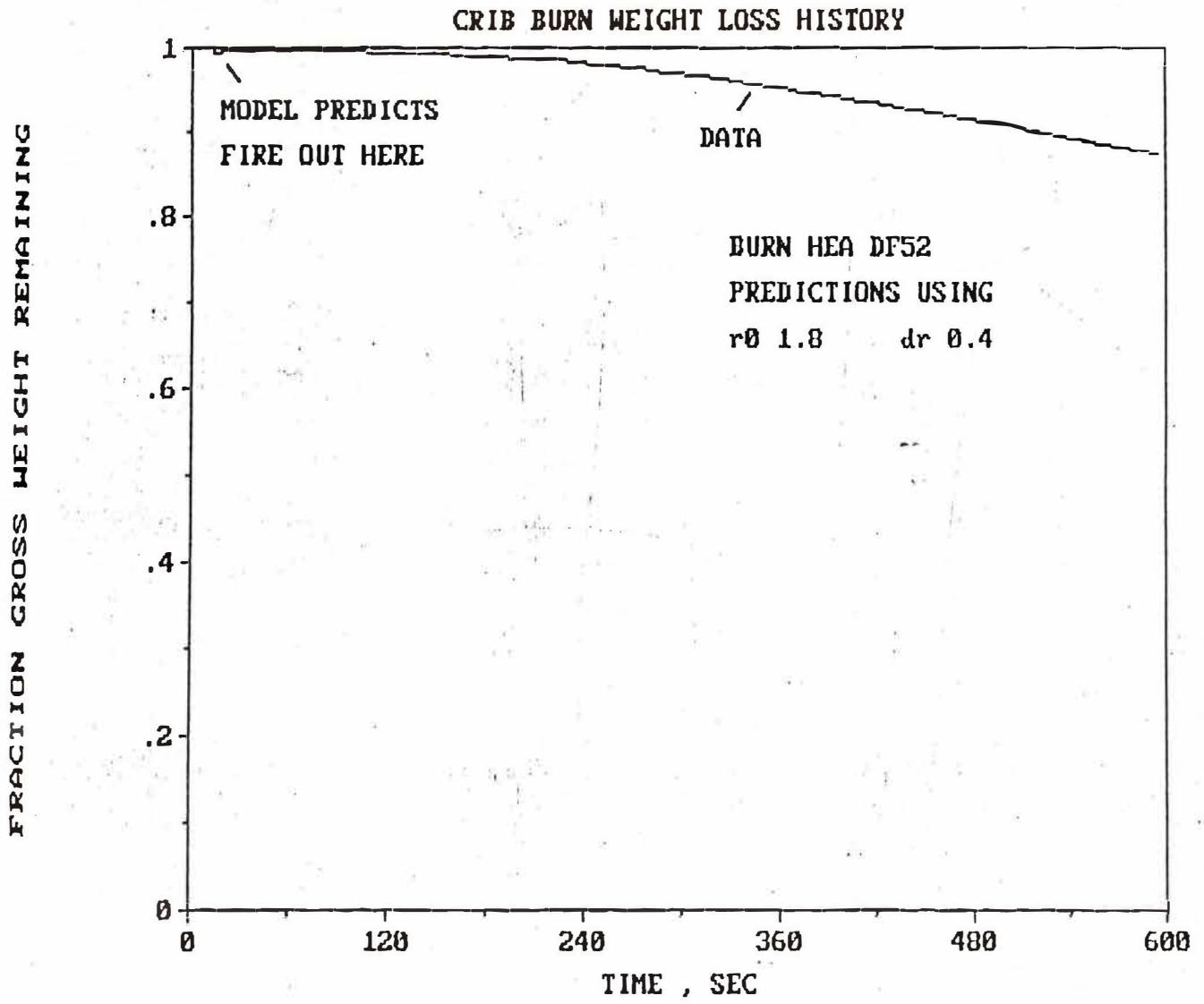

CRIB BURN WEIGHT LOSS HISTORY
FRACTION GROSS WEIGHT REMAINING
1
.8
.6
.4
.2
0
0
120
240
360
480
600
TIME , SEC
BURN HEA SF51
PREDICTIONS USING
r0 1.8 dr 0.4

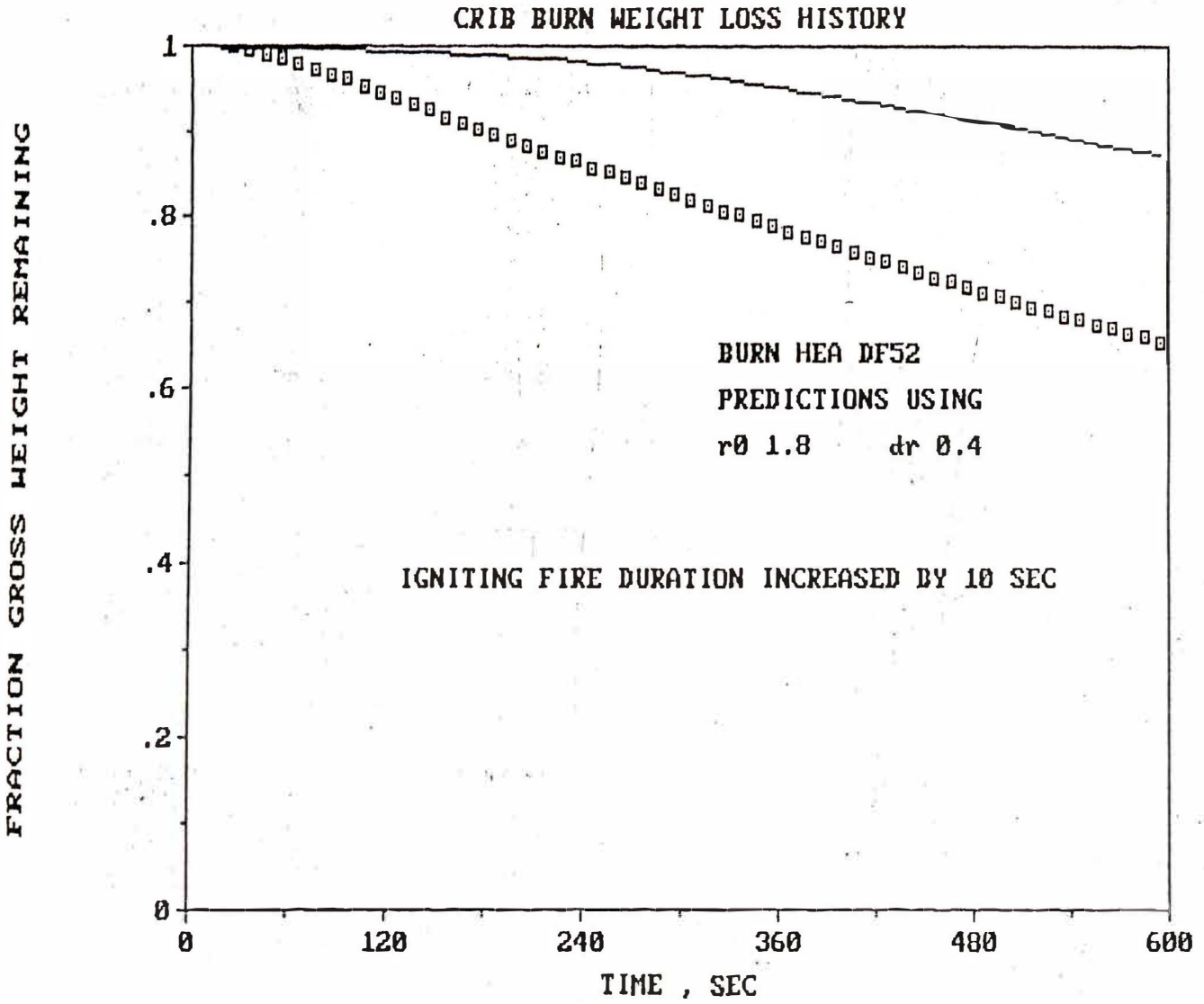

CRIB BURN WEIGHT LOSS HISTORY
FRACTION GROSS WEIGHT REMAINING
1
.8
.6
.4
.2
0
0
120
240
360
480
600
TIME , SEC
BURN HEA 3T151
PREDICTIONS USING
r0 1.8 dr 0.4

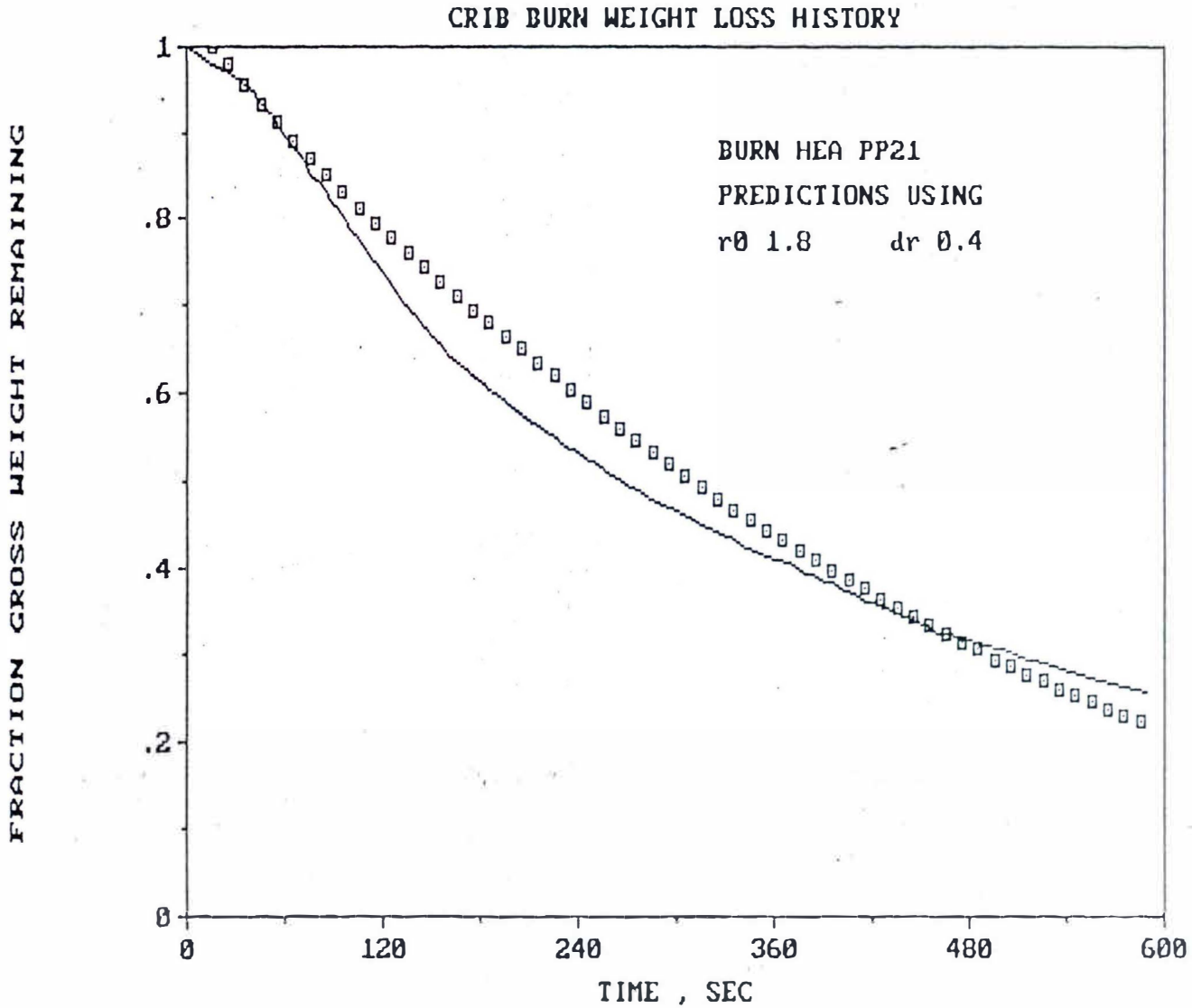

CRIB BURN WEIGHT LOSS HISTORY
FRACTION GROSS WEIGHT REMAINING
1
.8
.6
.4
.2
0
0
120
240
360
480
600
TIME , SEC
BURN HEA 3T161
PREDICTIONS USING
r0 1.8 dr 0.4

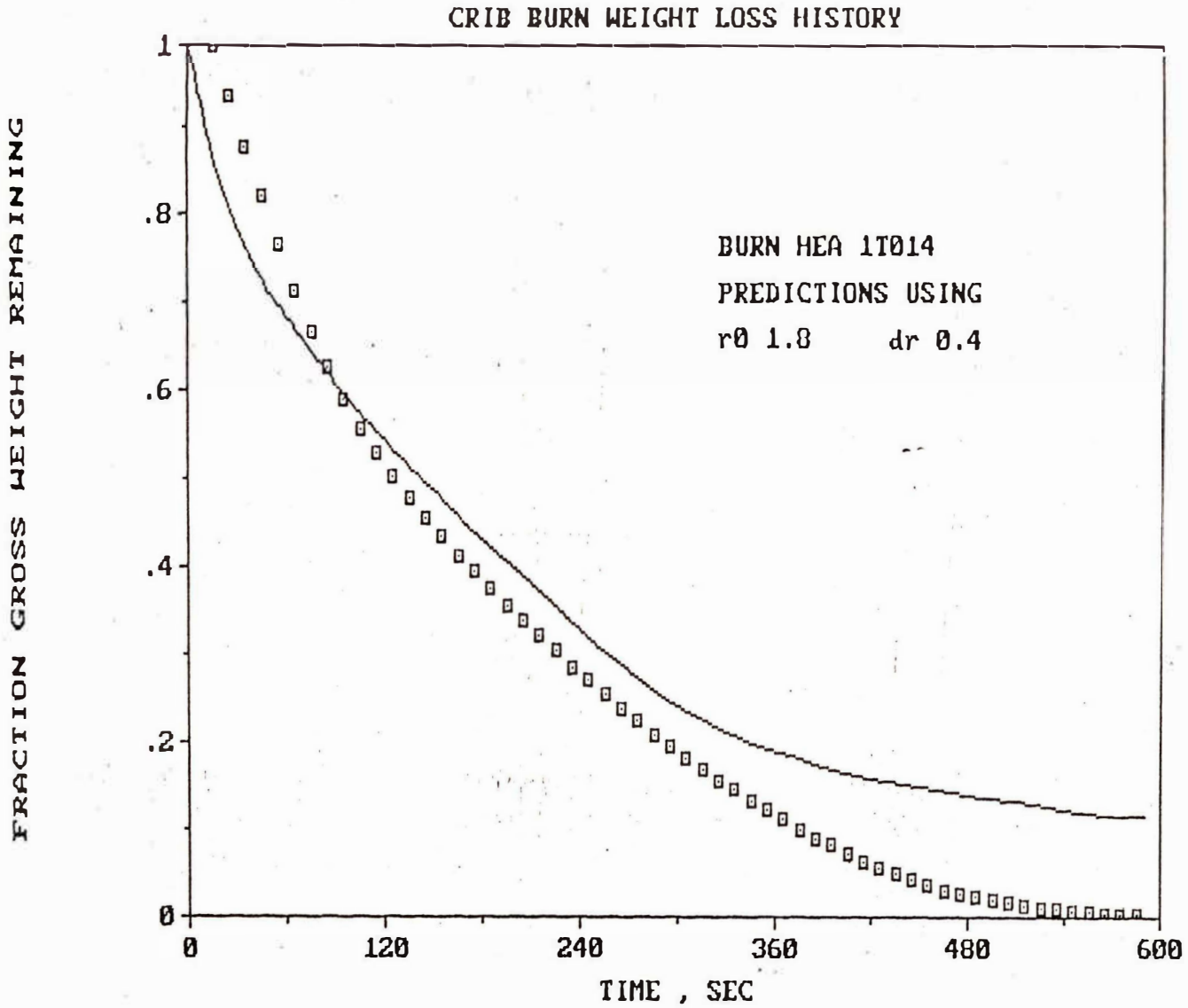

CRIB BURN WEIGHT LOSS HISTORY
BURN HEA DF31
PREDICTIONS USING
r0 1.8 dr 0.4
FRACTION GROSS WEIGHT REMAINING
1
.8
.6
.4
.2
0
0
120
240
360
480
600
TIME , SEC

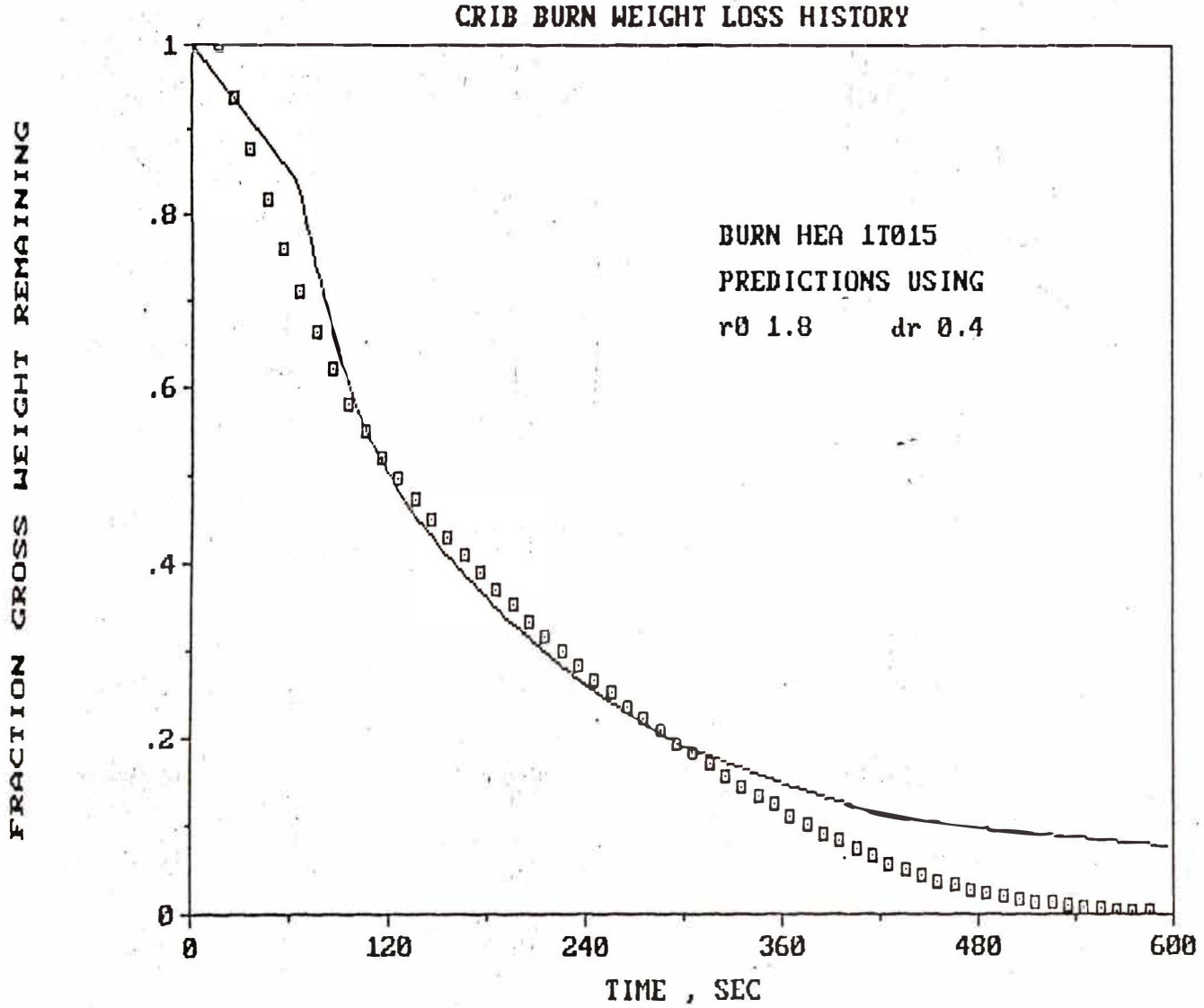

CRIB BURN WEIGHT LOSS HISTORY
FRACTION GROSS WEIGHT REMAINING
1
.8
.6
.4
.2
0
MODEL PREDICTS
FIRE OUT HERE
DATA
BURN HEA DF51
PREDICTIONS USING
r0 1.8 dr 0.4
0
120
240
360
480
600
TIME , SEC

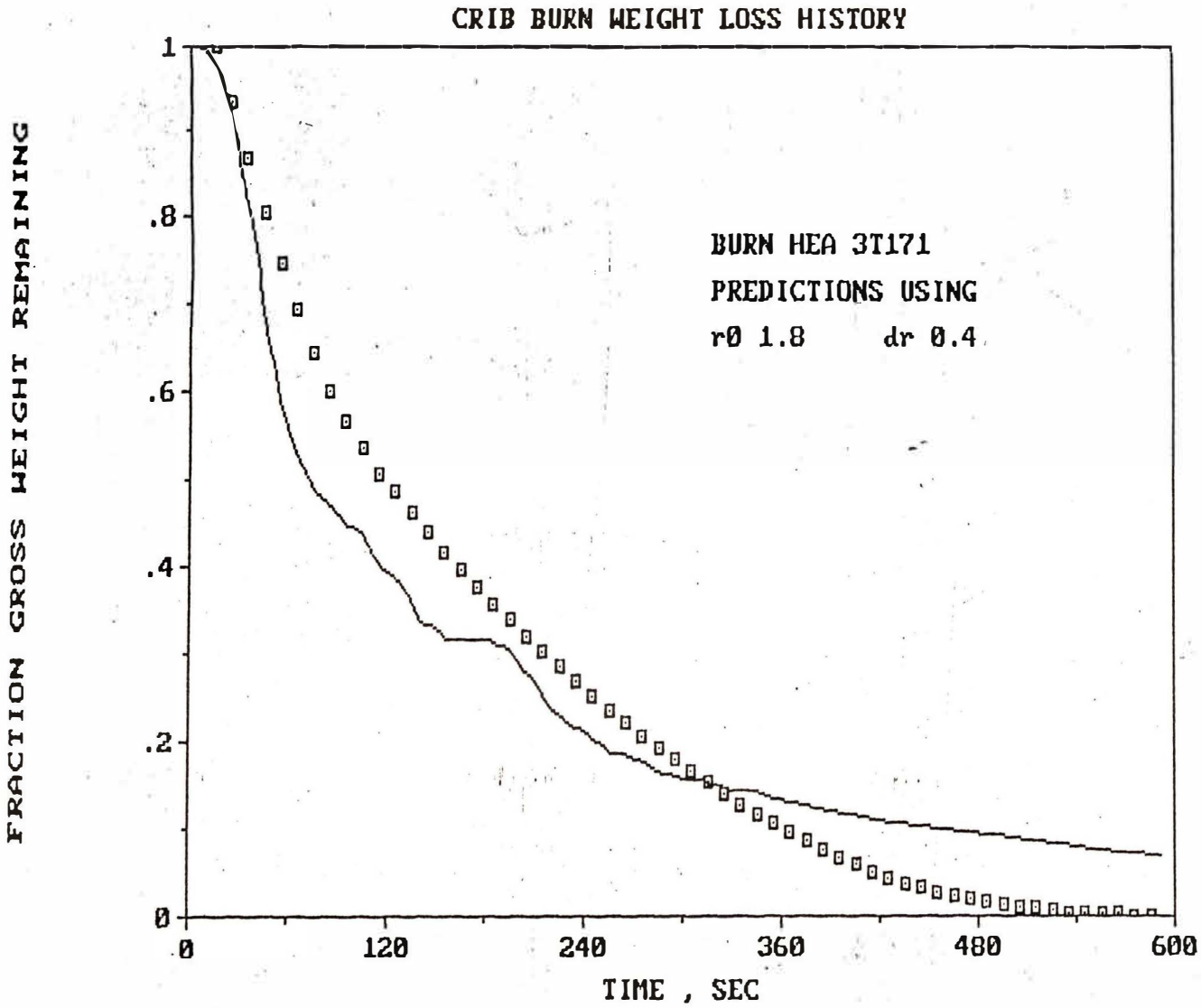

CRIB BURN WEIGHT LOSS HISTORY
FRACTION GROSS WEIGHT REMAINING
1
.8
.6
.4
.2
0
MODEL PREDICTS
FIRE OUT HERE
DATA
BURN HEA DF52
PREDICTIONS USING
r0 1.8 dr 0.4
0
120
240
360
480
600
TIME , SEC

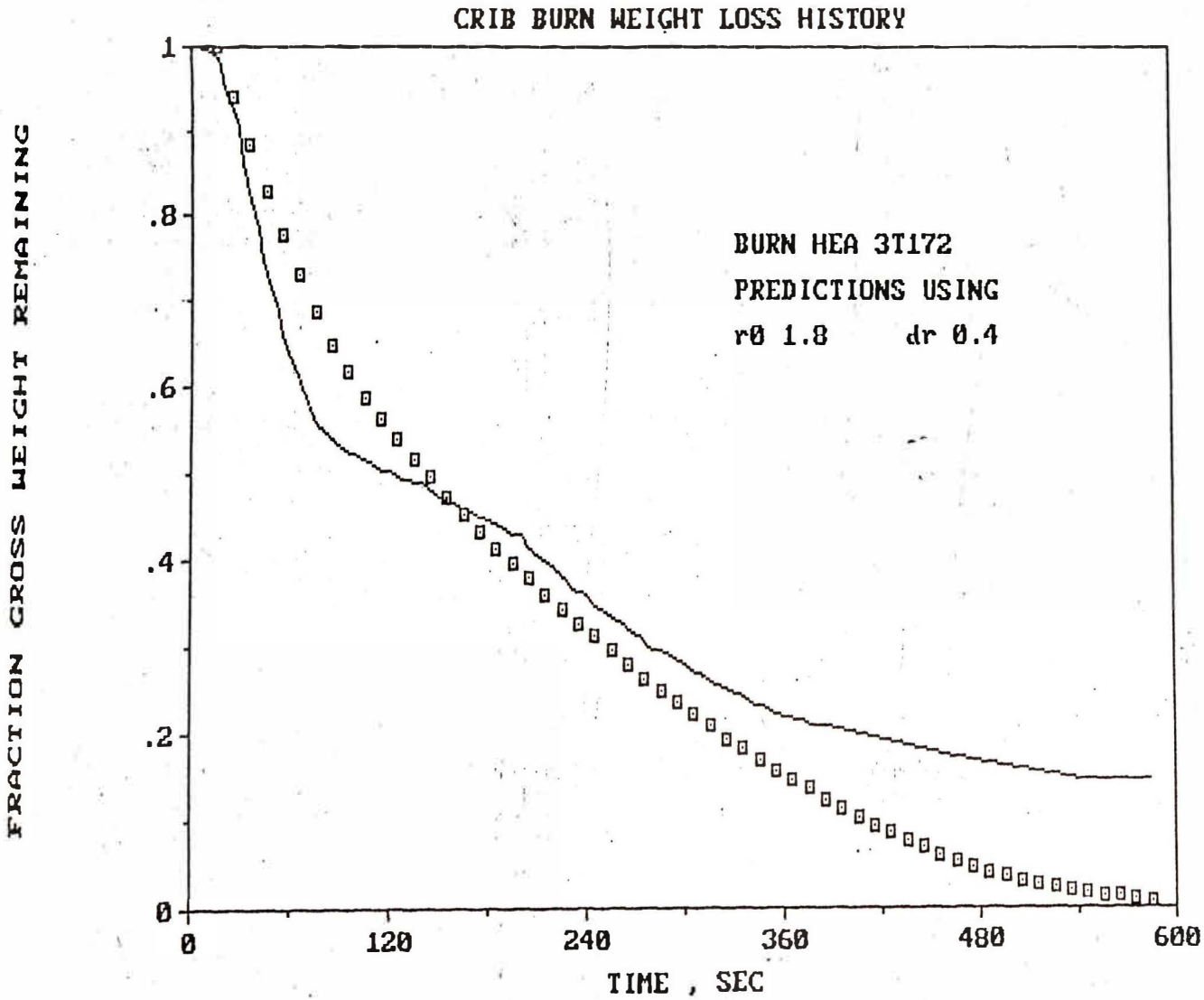

CRIB BURN WEIGHT LOSS HISTORY
FRACTION GROSS WEIGHT REMAINING
1
.8
.6
.4
.2
0
0
120
240
360
480
600
TIME , SEC
BURN HEA DF52
PREDICTIONS USING
r0 1.8 dr 0.4
IGNITING FIRE DURATION INCREASED BY 10 SEC

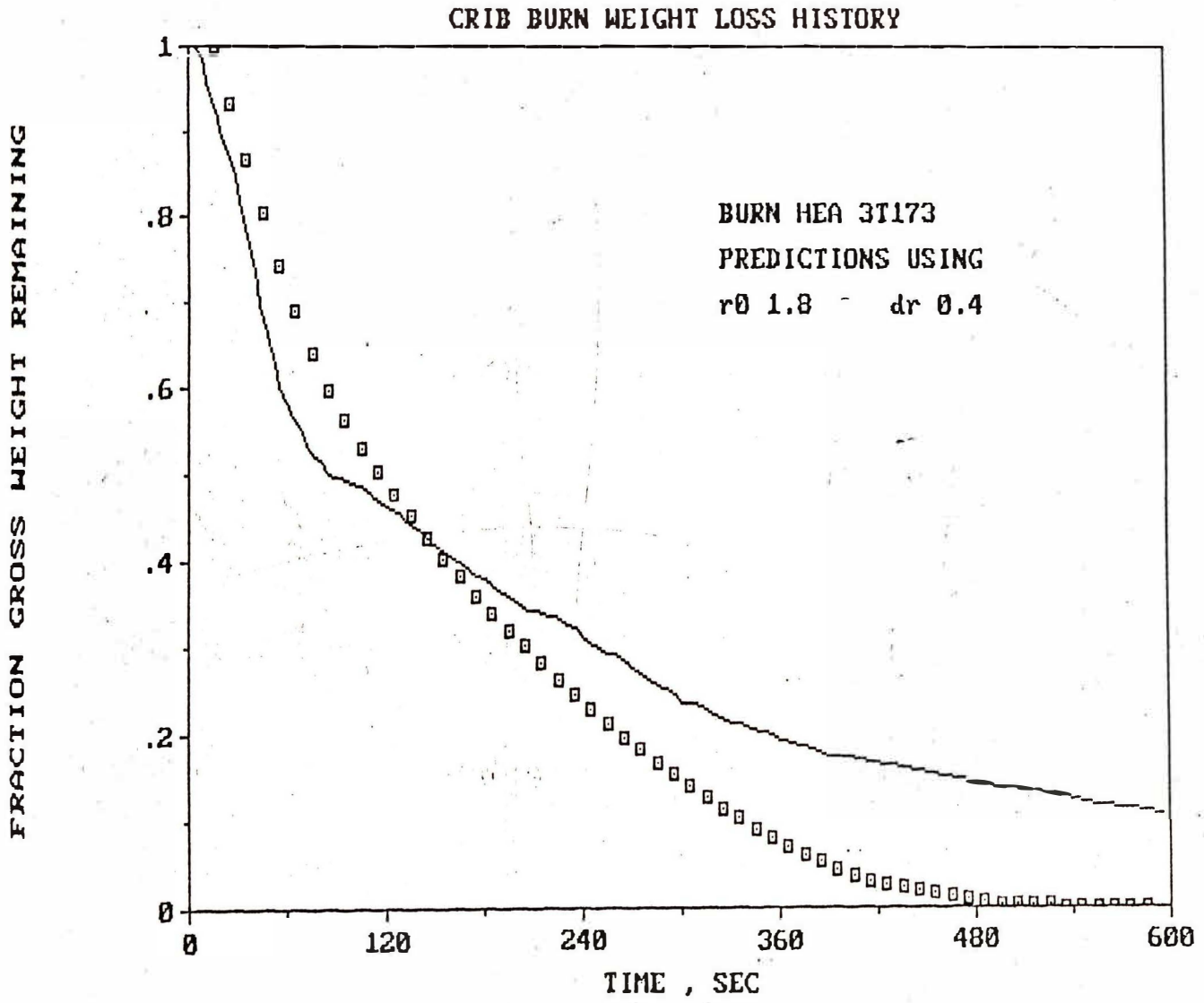

CRIB BURN WEIGHT LOSS HISTORY
FRACTION GROSS WEIGHT REMAINING
1
.8
.6
.4
.2
0
0
120
240
360
480
600
TIME , SEC
BURN HEA PP21
PREDICTIONS USING
r0 1.8
dr 0.4


CRIB BURN WEIGHT LOSS HISTORY
FRACTION GROSS WEIGHT REMAINING
1
.8
.6
.4
.2
0
0
120
240
360
480
600
TIME , SEC
BURN HEA 1T014
PREDICTIONS USING
r0 1.8
dr 0.4


CRIB BURN WEIGHT LOSS HISTORY
FRACTION GROSS WEIGHT REMAINING
1
.8
.6
.4
.2
0
0
120
240
360
480
600
TIME , SEC
BURN HEA 1T015
PREDICTIONS USING
r0 1.8 dr 0.4

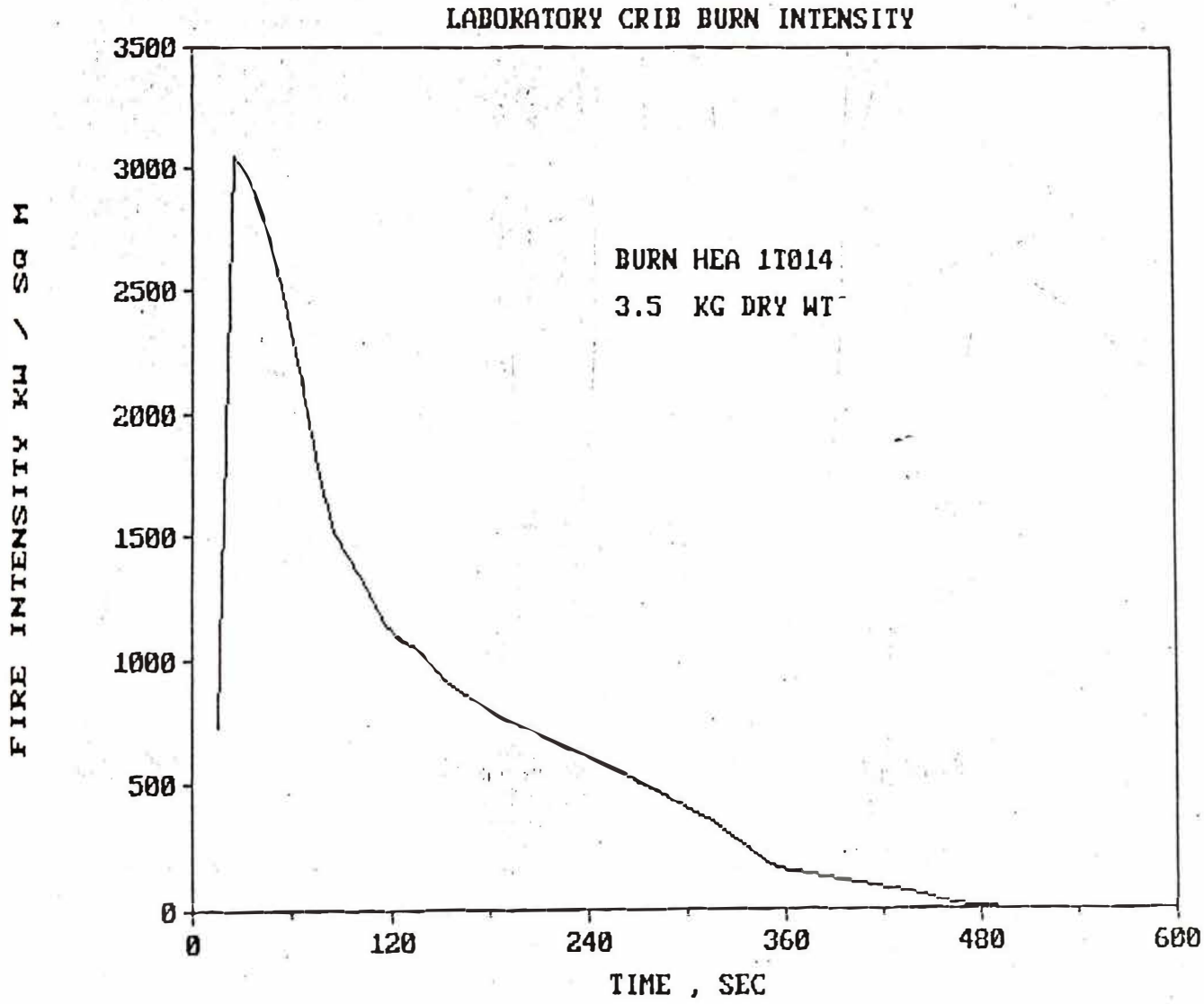

CRIB BURN WEIGHT LOSS HISTORY
FRACTION GROSS WEIGHT REMAINING
1
.8
.6
.4
.2
0
0
120
240
360
480
600
TIME , SEC
BURN HEA 3T171
PREDICTIONS USING
r0 1.8
dr 0.4

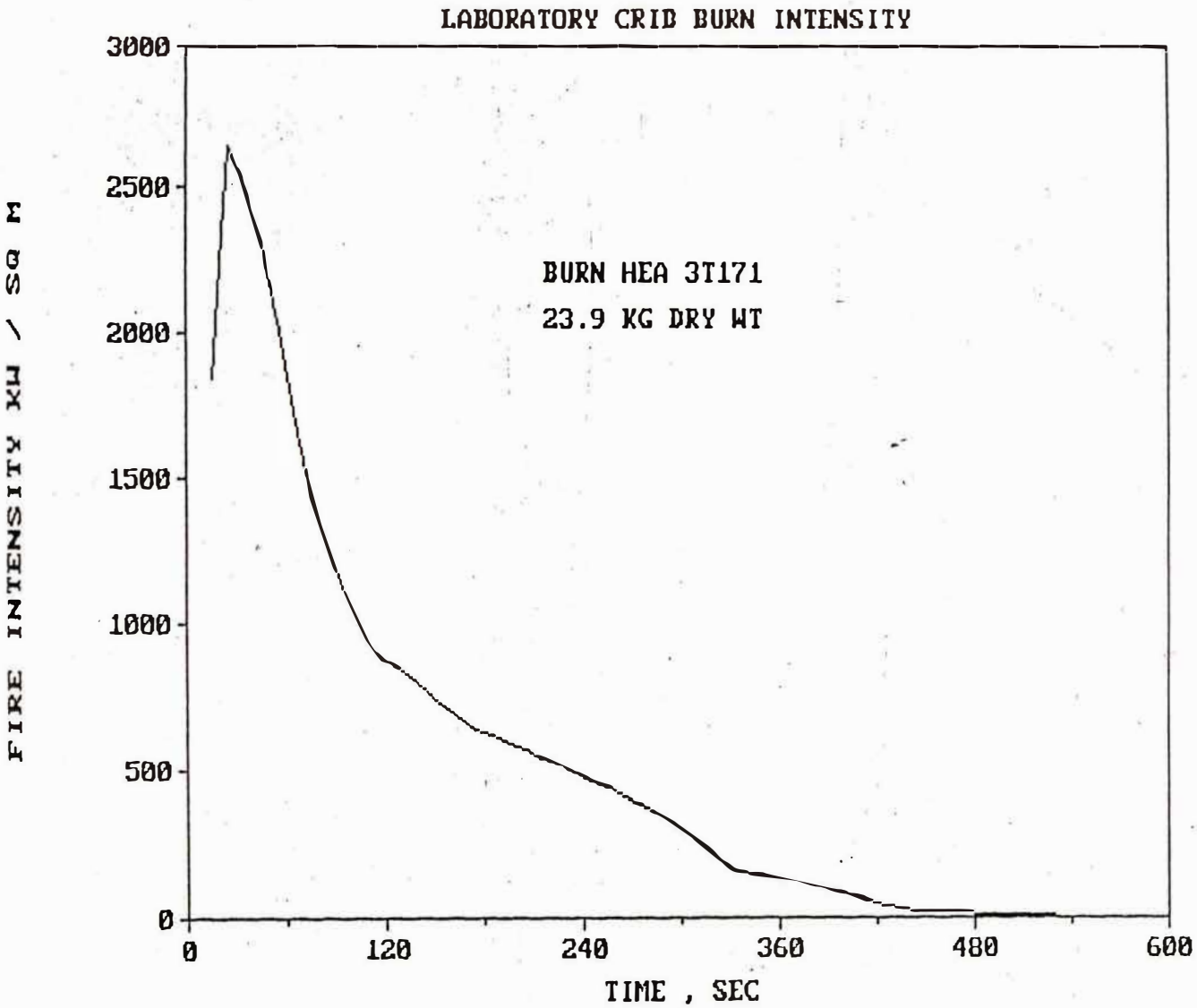

CRIB BURN WEIGHT LOSS HISTORY
FRACTION GROSS WEIGHT REMAINING
1
.8
.6
.4
.2
0
0
120
240
360
480
600
TIME , SEC
BURN HEA 3T172
PREDICTIONS USING
r0 1.8 dr 0.4

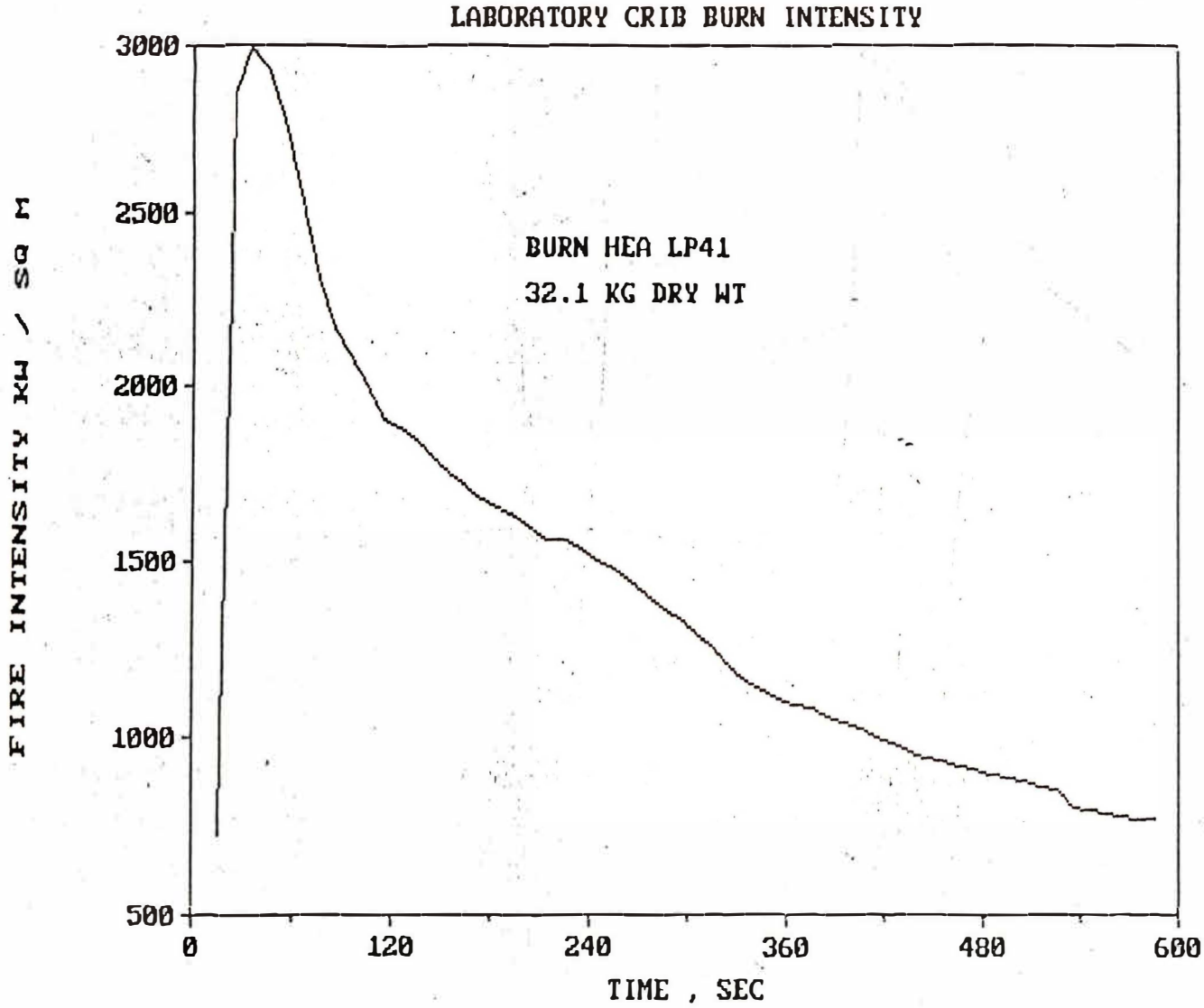

CRIB BURN WEIGHT LOSS HISTORY
FRACTION GROSS WEIGHT REMAINING
1
.8
.6
.4
.2
0
0
120
240
360
480
600
TIME , SEC
BURN HEA 3T173
PREDICTIONS USING
r0 1.8 - dr 0.4

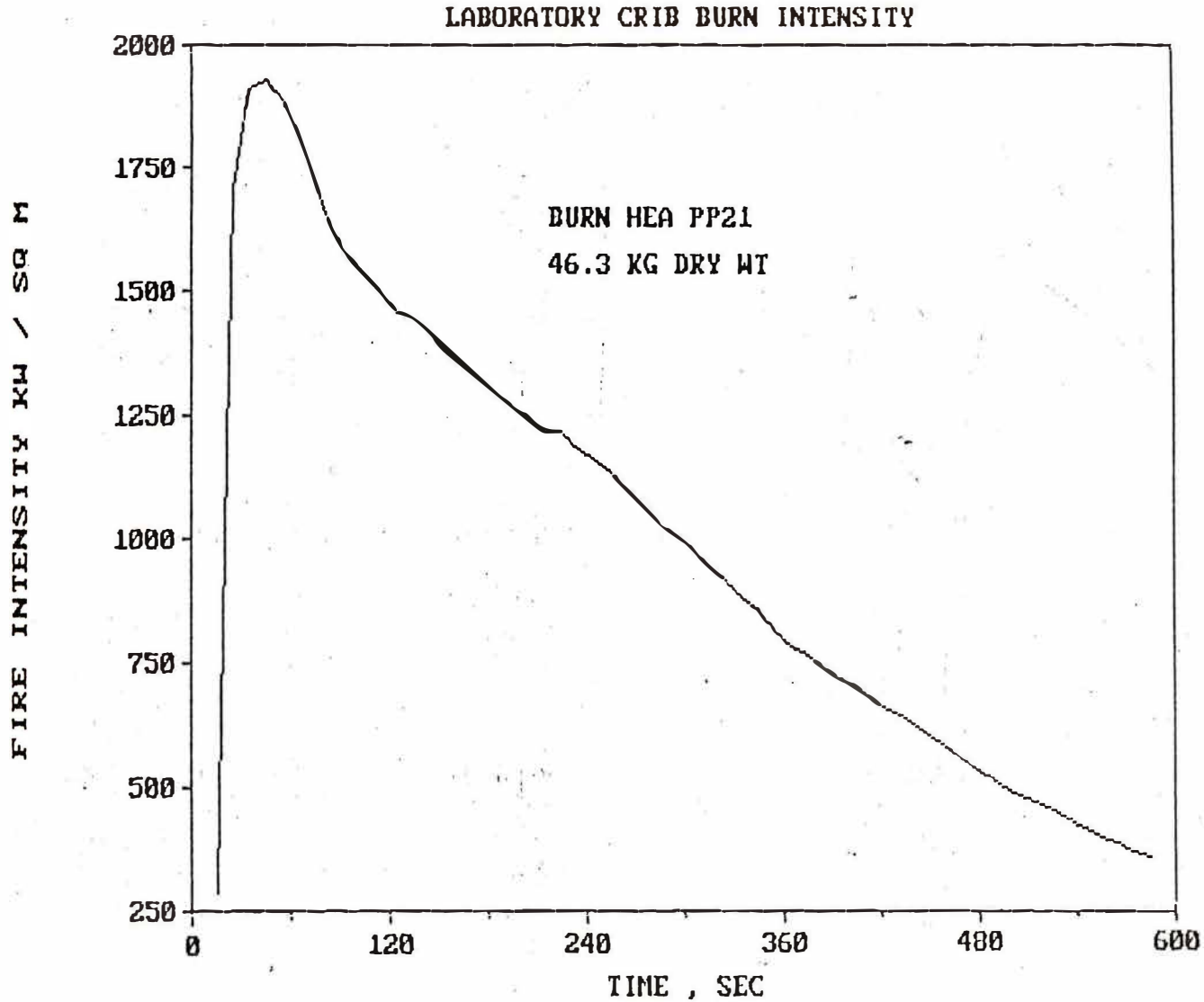
PREDICTED FIRE INTENSITIES VERSUS TIME

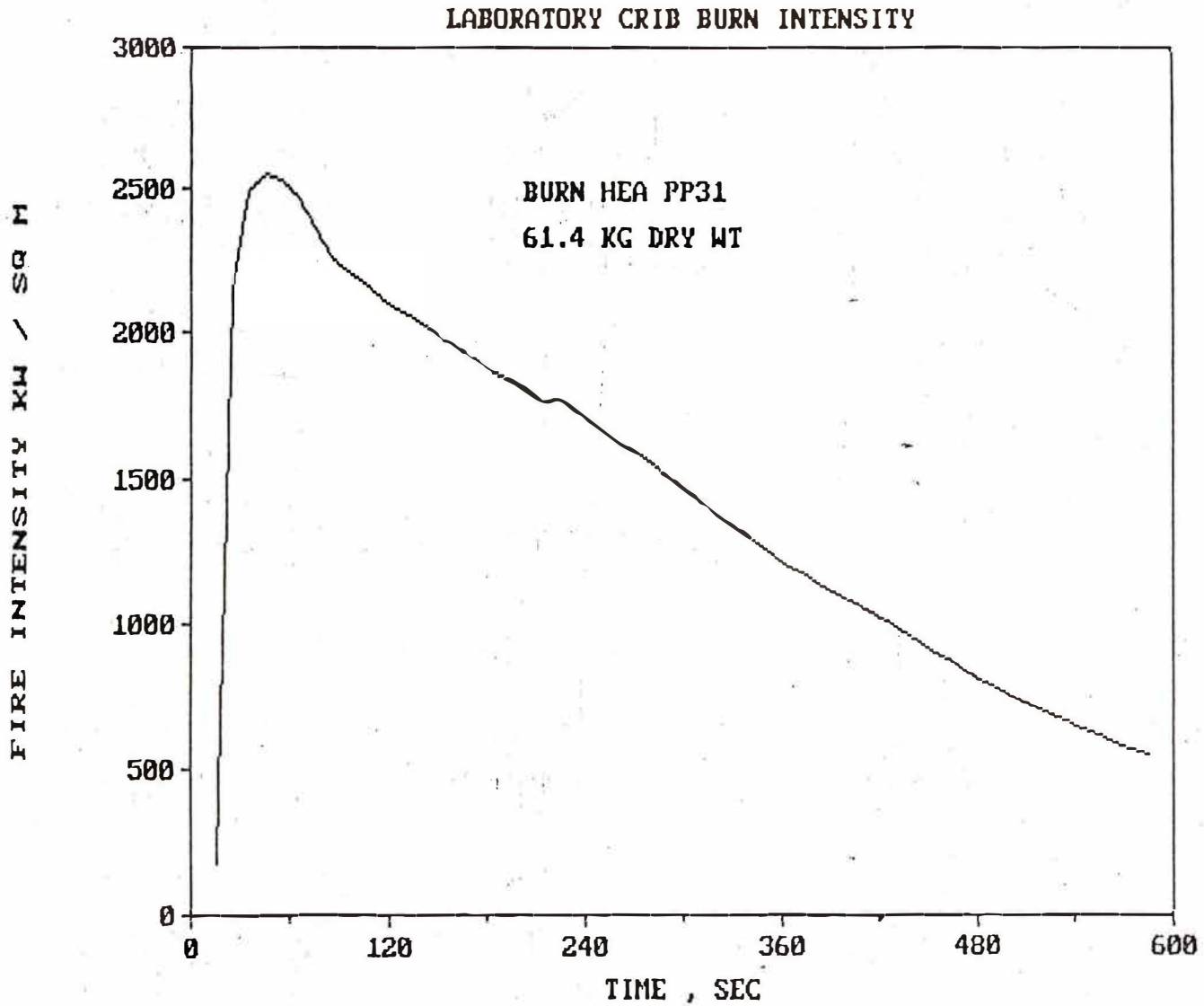
FOR A SAMPLE SET OF CRIB BURNS

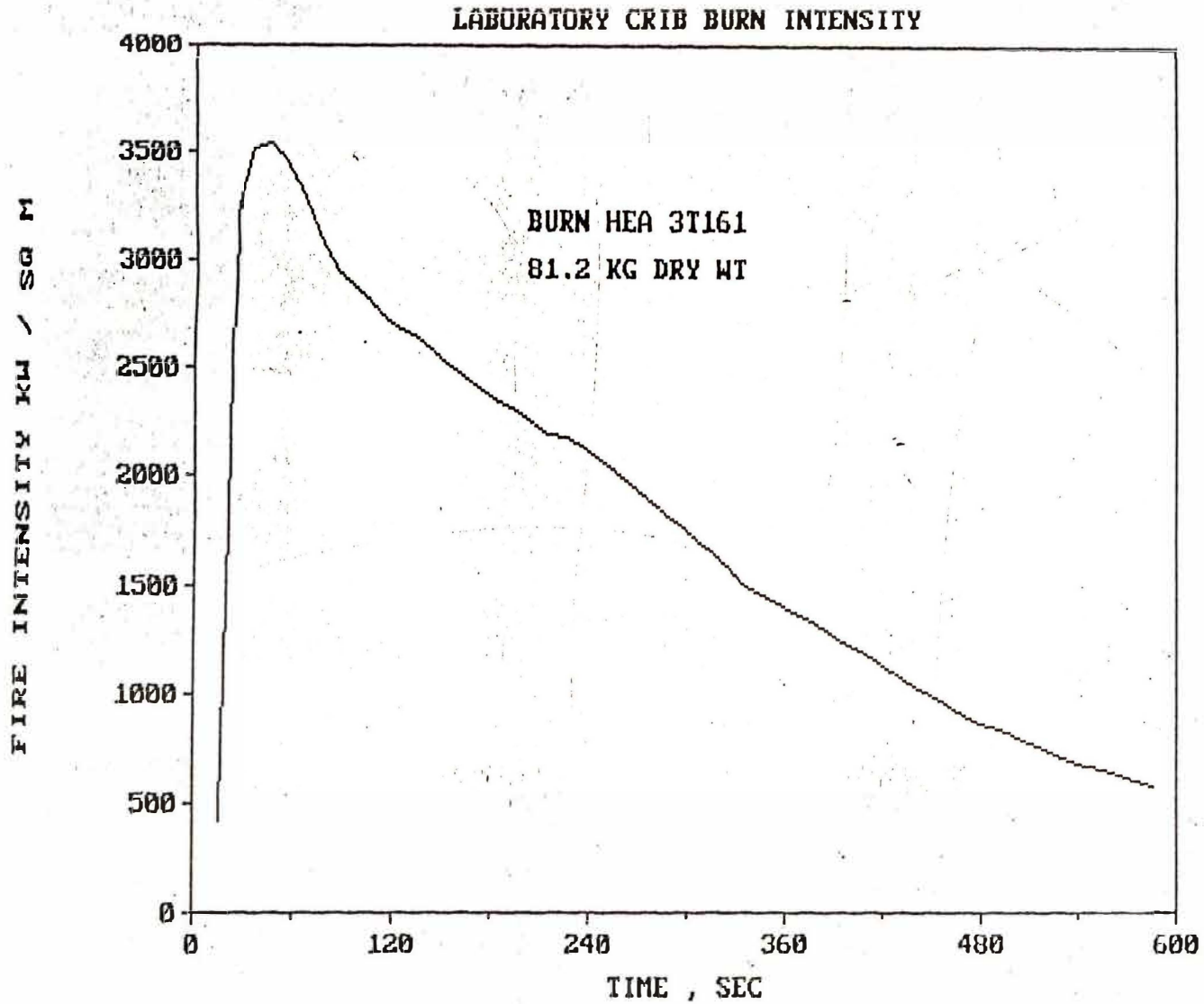

LABORATORY CRIB BURN INTENSITY
FIRE INTENSITY KW / SQ M
3500
3000
2500
2000
1500
1000
500
0
BURN HEA 1T014
3.5 KG DRY WT
0
120
240
360
480
600
TIME , SEC

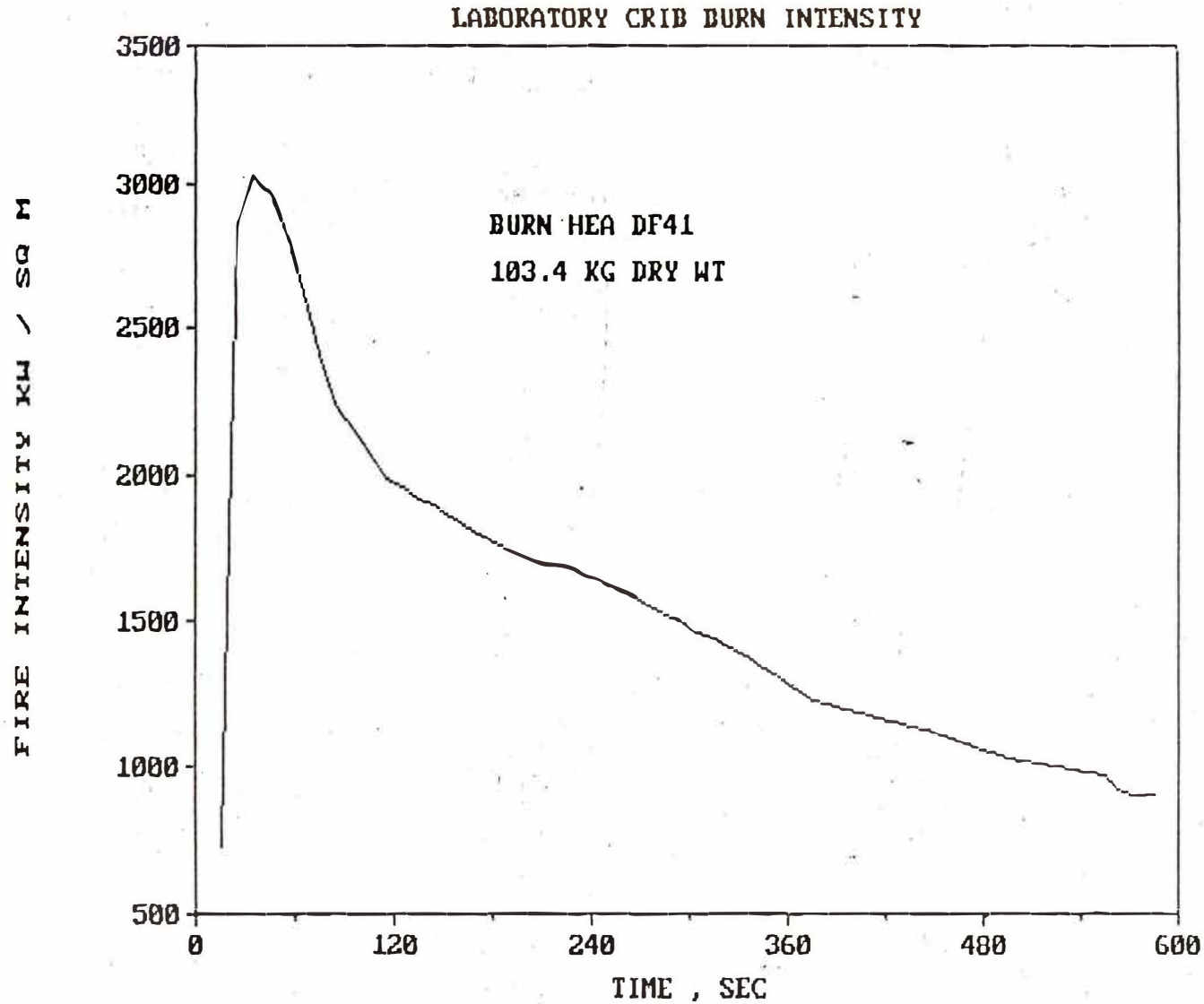

LABORATORY CRIB BURN INTENSITY
BURN HEA 3T171
23.9 KG DRY WT
FIRE INTENSITY KW / SQ M
TIME , SEC
3000
2500
2000
1500
1000
500
0
0
120
240
360
480
600

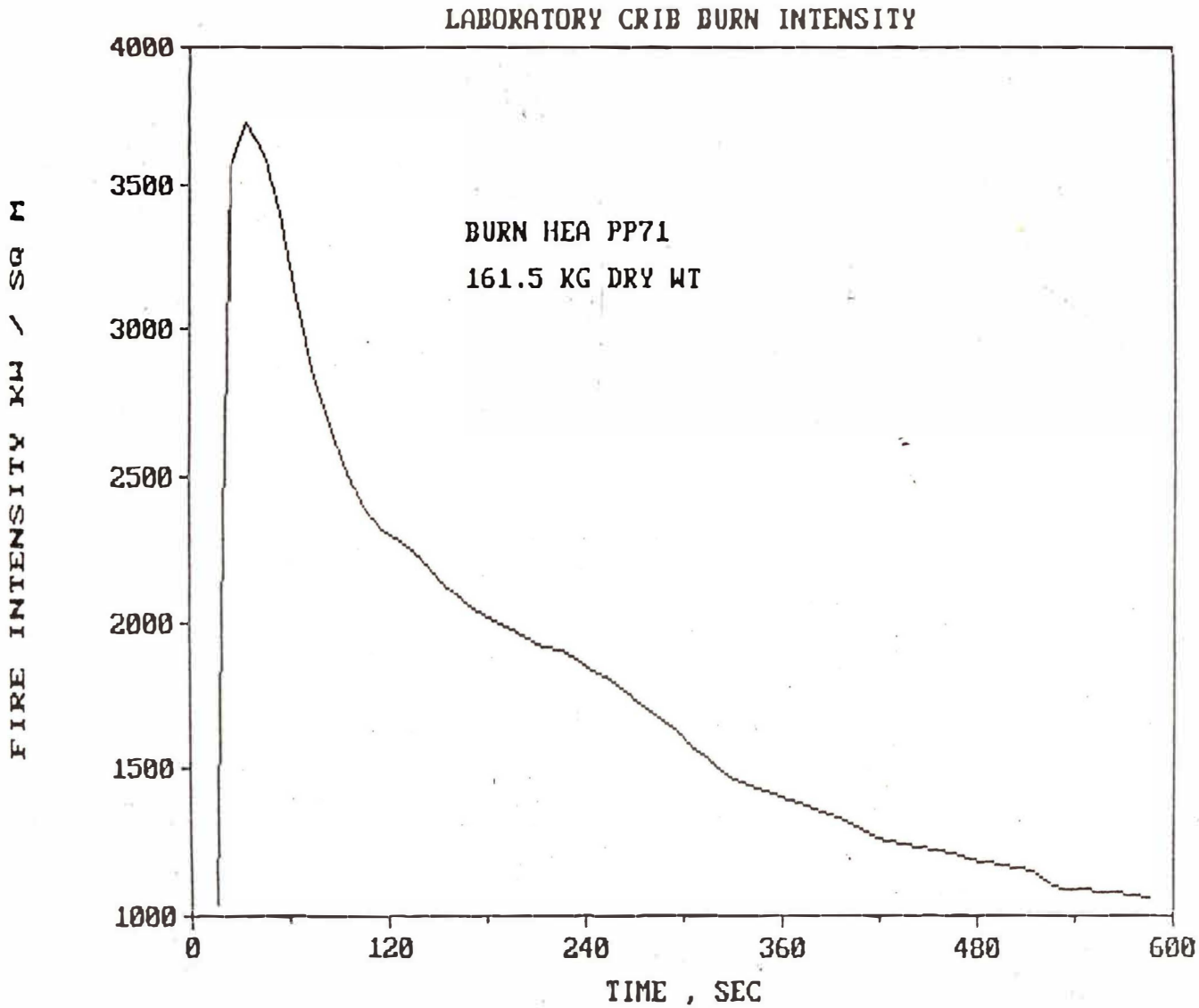

LABORATORY CRIB BURN INTENSITY
FIRE INTENSITY KW / SQ M
3000
2500
2000
1500
1000
500
BURN HEA LP41
32.1 KG DRY WT
0
120
240
360
480
600
TIME , SEC

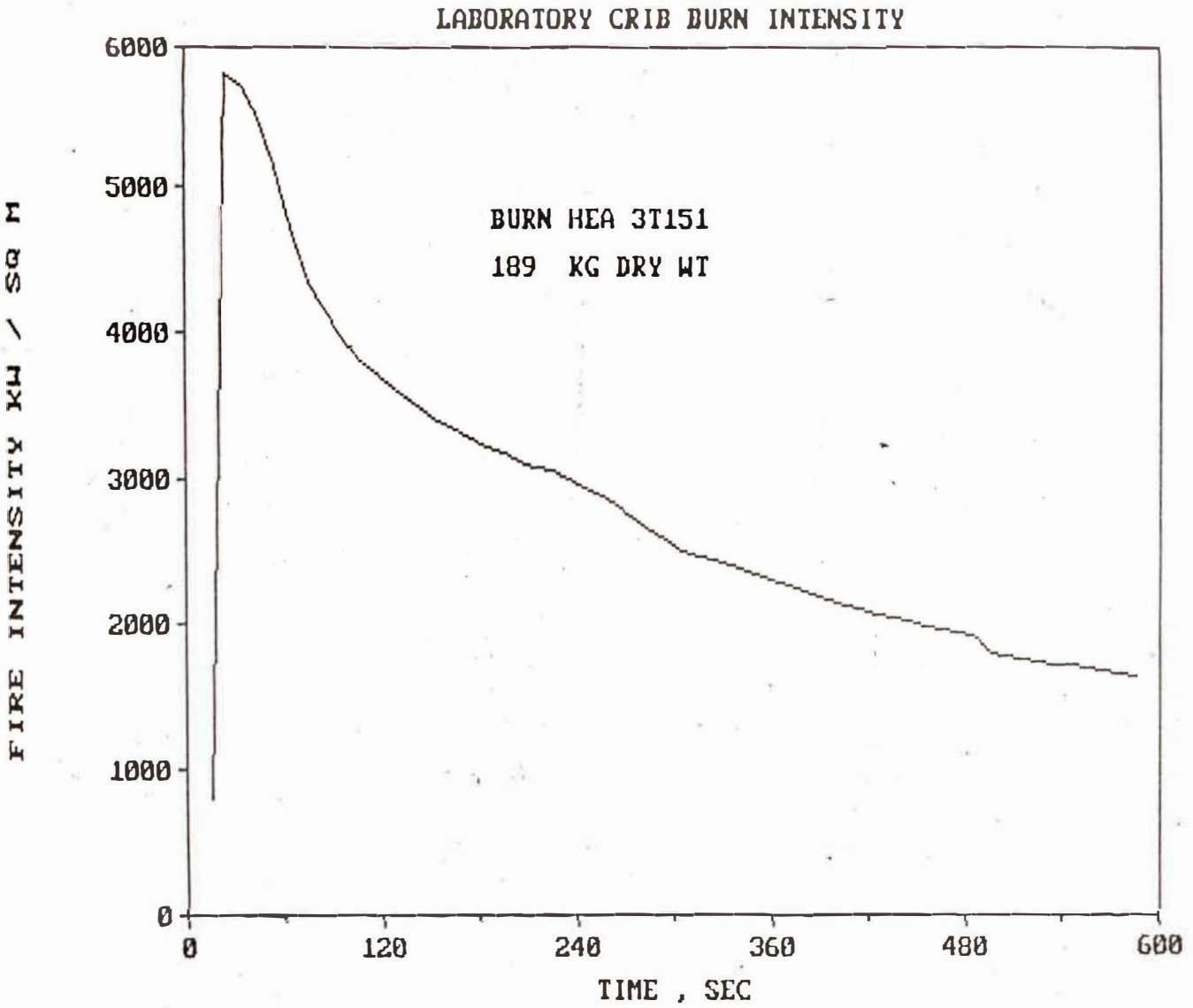

LABORATORY CRIB BURN INTENSITY
BURN HEA PP21
46.3 KG DRY WT
FIRE INTENSITY KW / SQ M
2000
1750
1500
1250
1000
750
500
250
0
120
240
360
480
600
TIME , SEC

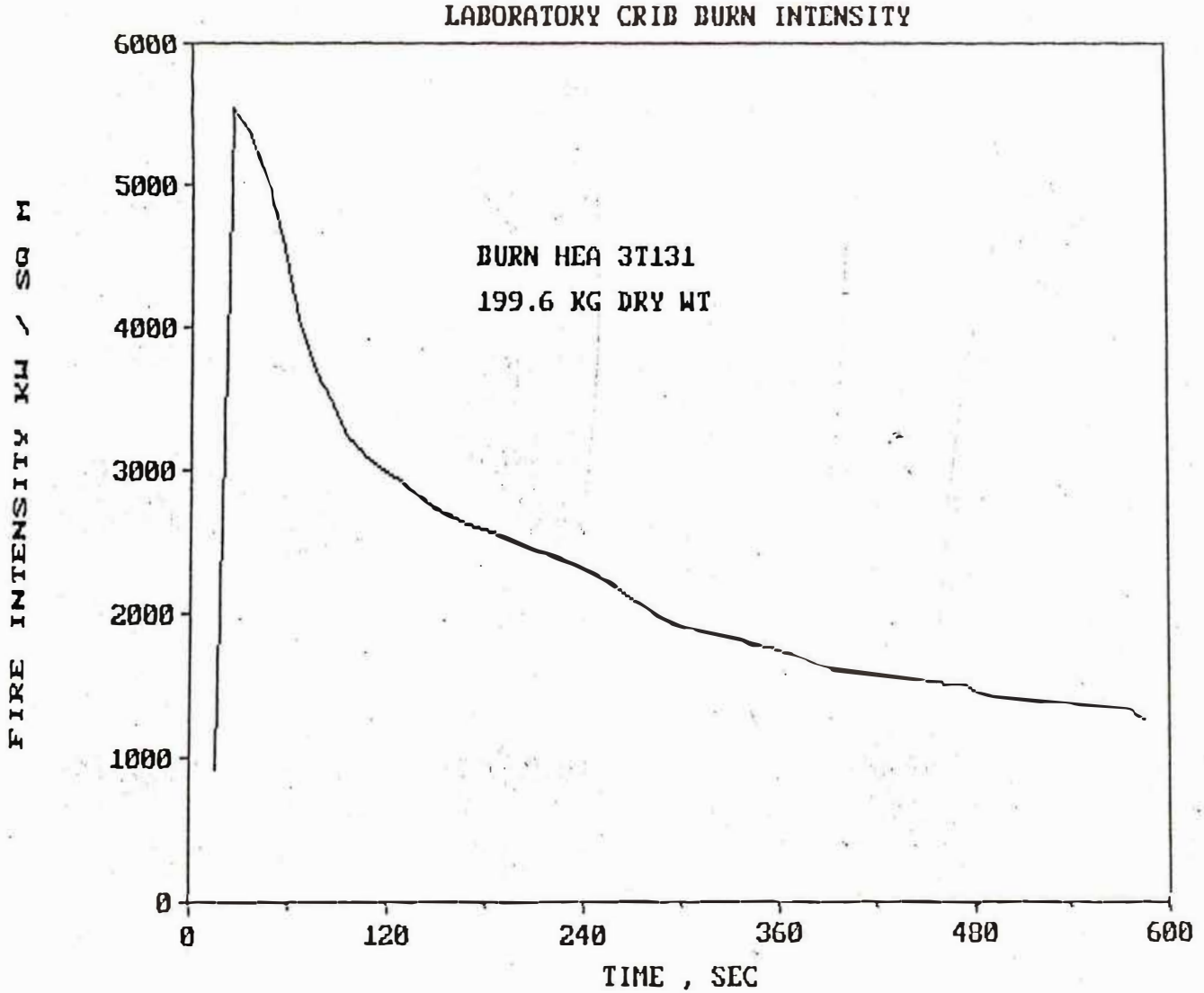

LABORATORY CRIB BURN INTENSITY
BURN HEA PP31
61.4 KG DRY WT
FIRE INTENSITY KW / SQ M
3000
2500
2000
1500
1000
500
0
0
120
240
360
480
600
TIME , SEC


LABORATORY CRIB BURN INTENSITY
BURN HEA 3T161
81.2 KG DRY WT
FIRE INTENSITY KW / SQ M
TIME , SEC
4000
3500
3000
2500
2000
1500
1000
500
0
0
120
240
360
480
600


LABORATORY CRIB BURN INTENSITY
BURN HEA DF41
103.4 KG DRY WT
FIRE INTENSITY KW / SQ M
3500
3000
2500
2000
1500
1000
500
0
120
240
360
480
600
TIME , SEC


LABORATORY CRIB BURN INTENSITY
BURN HEA PP71
161.5 KG DRY WT
FIRE INTENSITY KW / SQ M
4000
3500
3000
2500
2000
1500
1000
0
120
240
360
480
600
TIME , SEC


LABORATORY CRIB BURN INTENSITY
BURN HEA 3T151
189 KG DRY WT
FIRE INTENSITY KW / SQ M
6000
5000
4000
3000
2000
1000
0
0
120
240
360
480
600
TIME , SEC


LABORATORY CRIB BURN INTENSITY
BURN HEA 3T131
199.6 KG DRY WT
FIRE INTENSITY KW / SQ M
6000
5000
4000
3000
2000
1000
0
0
120
240
360
480
600
TIME , SEC

APPENDIX B. SOURCE CODE LISTING

In this appendix the FORTRAN source code for the BURNUP model is given. The various subroutines are listed separately, in approximately the order in which they are encountered as the model is exercised. The order of appearance of the elements is:

1. Program BURNUP, driver program for the model.

   1.1 Subroutine DUFBRN, appended to BURNUP, models duff burning.

2. Subroutine GETDAT for input, modification, and archiving of data.

   2.1 Subroutine RETRY, appended to GETDAT, checks user input.

3. Subroutine ARRAYS orders data, generates interaction statistics.

4. Subroutine SORTER performs fuel ordering sort.

5. Subroutine OVLAPS does interaction computations.

6. Subroutine START applies ignition heat pulse, begins simulation.

7. Subroutine FIRINT computes overall and near-fuel fire intensity.

8. Subroutine STASH stores fire history data.

9. Subroutine TIGNIT computes halfspace surface ignition time under steady radiant heating with surface film cooling.

10. Subroutine DRYTIM computes time required for surface to reach drying temperature for halfspace under steady radiant heating with surface film cooling.

11. Subroutine HEATX computes convective and radiation-adjusted film heat transfer coefficients, and Biot number, for a cylinder in a fire environment specified by a given temperature.

12. Function TEMPF gives a local fire environment temperature.

13. Subroutine STEP implements one time step in the simulation.

14. Subroutine SUMMARY archives a summary of fuel consumption by size class, in one of two levels of detail.

Note that the maximum list length for fuel components is given by the first statement in program BURNUP, a parameter specification. This is the only statement that need be changed in order to shrink or expand the maximum length of the list of fuel components.

```
      program BURNUP

      parameter ( MAXNO = 10 )
      parameter ( MAXKL = MAXNO * ( MAXNO + 1 ) / 2 + MAXNO )
      parameter ( MXSTEP = 20 )

      real*4 wdry( maxno ) , ash( maxno ) , htval( maxno )
      real*4 fmois( maxno ) , dendry( maxno ) , sigma( maxno )
      real*4 cheat( maxno ) , condry( maxno ) , alfa( maxno )
      real*4 tpig( maxno ) , tchar( maxno )
      real*4 flit( maxno ) , fout( maxno ) , work( maxno )
      real*4 elam( maxno , maxno ) , alone( maxno )
      real*4 area( maxno ) , fint( maxno )
      real*4 xmat( maxkl ) , tdry( maxkl )
      real*4 tign( maxkl ) , tout( maxkl )
      real*4 wo( maxkl ) , wodot( maxkl )
      real*4 diam( maxkl ) , ddot( maxkl )
      real*4 qcum( maxkl ) , tcum( maxkl )
      real*4 acum( maxkl ) , qdot( maxkl , mxstep )
      integer key( maxno )
      character*12 parts( maxno ) , list( maxno ) , infile , outfil
      logical nohist

      fimin = 0.1
      nruns = 0

10    call GETDAT( infile , outfil , parts , wdry , ash , htval ,
     +             fmois , dendry , sigma , cheat , condry , tpig ,
     +             tchar , number , maxno , fi , ti , u , d , tpamb ,
     +             ak , r0 , dr , dt , ntimes , wdf , dfm , nruns ,
     +             area , fint )

11    write( * , 12 )
12    format(' Enter 1 to store fire history, 0 to skip    ',$)
      read( * , * ) ihist
      nohist = ( ihist .EQ. 0 )
      if( ( ihist .NE. 1 ) .AND. ( .NOT. nohist ) ) goto 11

      call ARRAYS( maxno , number , wdry , ash , dendry , fmois ,
     +             sigma , htval , cheat , condry , tpig , tchar ,
     +             diam , key , work , ak , elam , alone , xmat , wo ,
     +             maxkl , parts , list , area )

      now = 1
      tis = ti
      call DUFBRN( wdf , dfm , dfi , tdf )

      call START( tis , mxstep , now , maxno , number , wo , alfa ,
     +            dendry , fmois , cheat , condry , diam , tpig ,
     +            tchar , xmat , tpamb , tpdry , fi , flit , fout ,
     +            tdry , tign , tout , qcum , tcum , acum , qdot ,
     +            ddot , wodot , work , u , d , r0 , dr , ch2o ,
     +            ncalls , maxkl )

      if( tis .LT. tdf ) then
```

```
      fid = dfi
    else
      fid = 0.
    end if

    if( nohist ) goto 14

    call STASH( tis , now , maxno , number , outfil , fi , flit ,
   +            fout , wo , wodot , diam , ddot , tdry , tign ,
   +            tout , fmois , maxkl , nun )

14  call FIRINT( wodot , ash , htval , maxno , number , maxkl , area ,
   +             fint , fi )

    if( fi .LE. fimin ) goto 22

20  call STEP( dt , mxstep , now , maxno , number , wo , alfa ,
   +           dendry , fmois , cheat , condry , diam , tpig ,
   +           tchar , xmat , tpamb , tpdry , fi , flit , fout ,
   +           tdry , tign , tout , qcum , tcum , acum , qdot ,
   +           ddot , wodot , work , u , d , r0 , dr , ch2o ,
   +           ncalls , maxkl , tis , fint , fid )

    now = now + 1
    tis = tis + dt
    if( tis .LT. tdf ) then
      fid = dfi
    else
      fid = 0.
    end if

    call FIRINT( wodot , ash , htval , maxno , number , maxkl , area ,
   +             fint , fi )

    if( fi .LE. fimin ) goto 22

    if( nohist ) goto 21

    call STASH( tis , now , maxno , number , outfil , fi , flit ,
   +            fout , wo , wodot , diam , ddot , tdry , tign ,
   +            tout , fmois , maxkl , nun )

21  continue

    if( now .LT. ntimes ) goto 20

22  close( nun )

    outfil = 'SUMMARY.DAT'
    call SUMMARY( outfil , number , maxno , maxkl , parts , nun ,
   +tis , ak , wdry , fmois , sigma , tign , tout , xmat , wo , diam )
    close( nun )
25  write(*,30)
30  format(' Exercise completed.  Do another = 1 , quit = 0    ',$)
    read(*,*,err=25) next
```

```
if( next .EQ. 0 ) then
   close( 66 )
   stop' Terminated'
end if
if( next .NE. 1 ) goto 25
close( 66 )
goto 10

end


subroutine DUFBRN( wdf , dfm , dfi , tdf )

Duff burning rate (ergo, intensity) and duration

dfi = 0.
tdf = 0.
if( ( wdf .LE. 0. ) .OR. ( dfm .GE. 1.96 ) ) return
dfi = 11.25 - 4.05 * dfm
ff =  0.837 - 0.426 * dfm
tdf = 1.e+04 * ff * wdf / ( 7.5 - 2.7 * dfm )
return
end
```

```
      subroutine GETDAT( infile , outfil , parts , wdry , ash , htval ,
     +                   fmois , dendry , sigma , cheat , condry , tpig ,
     +                   tchar , number , maxno , fi , ti , u , d , tpamb ,
     +                   ak , r0 , dr , dt , ntimes , wdf , dfm , nruns ,
     +                   area , fint )

This routine prompts user for input data in the form of file names
or the direct input of numerical quantities.
Routine makes consistency checks, tests nominal quantities, and
allows review / revision of data before starting program,
and allows archiving of run conditions for future use.

      real*4 wdry( maxno ) , ash( maxno ) , htval( maxno )
      real*4 fmois( maxno ) , dendry( maxno ) , sigma( maxno )
      real*4 cheat( maxno ) , condry( maxno )
      real*4 tpig( maxno ) , tchar( maxno )
      real*4 area( maxno ) , fint( maxno )
      character*12 parts( maxno ) , infile , outfil , name1
      logical test

      data small / 1.e-06 / , big / 1.e+06 /
      data ash1 / 0.0001 / , ash2 / 0.1 /
      data htv1 / 1.0e+07 / , htv2 / 3.0e+07 /
      data fms1 / 0.01 / , fms2 / 3.0 /
      data den1 / 200.0 / , den2 / 1000.0 /
      data sig1 / 4.0 / , sig2 / 1.0e+04 /
      data cht1 / 1000.0 / , cht2 / 2000.0 /
      data con1 / 0.025 / , con2 / 0.25 /
      data tig1 / 200.0 / , tig2 / 400.0 /
      data tch1 / 250.0 / , tch2 / 500.0 /
      data fir1 / 100. / , fir2 / 1.0e+05 /
      data ti1 / 10.0 / , ti2 / 200.0 /
      data u1 / 0.0 / , u2 / 5.0 /
      data d1 / 0.1 / , d2 / 2.0 /
      data tam1 / -40.0 / , tam2 / 40.0 /
      data wdf1 / 0.1 / , wdf2 / 30.0 /
      data dfm1 / 0.1 / , dfm2 / 1.972 /

      if( nruns .GT. 0 ) goto 01
      do i = 1 , maxno
         wdry( i ) = 0.
         ash( i ) = 0.
         htval( i ) = 0.
         fmois( i ) = 0.
         dendry( i ) = 0.
         sigma( i ) = 0.
         cheat( i ) = 0.
         condry( i ) = 0.
         tpig( i ) = 0.
         tchar( i ) = 0.
         area( i ) = 0.
         fint( i ) = 0.
         parts( i ) = 'No data for'
      end do
      number = 0
```

```
      fi = 0.
      ti = 0.
      u = 0.
      d = 0.
      tpamb = 0.
      ak = 0.
      r0 = 0.
      dr = 0.
      dt = 0.
      ntimes = 0
      wdf = 0.
      dfm = 2.

 01   write(*,02)
 02   format(' Enter 1 for keyboard entry of data'/
     +       ' Enter 2 to fetch data stored in files'/
     +       ' Enter 0 to terminate program now   ',$)
      read(*,*,err=01) in
      if( in .EQ. 0 ) stop' Program ended'
      if( in .EQ. 2 ) goto 1000
      if( in .NE. 1 ) goto 01

      nruns = nruns + 1

 03   write(*,04) maxno
 04   format(' How many [ no more than'i3' ] fuel components ?   ',$)
      read(*,*,err=03) numb
      if( ( numb .LE. 0 ) .OR. ( numb .GT. maxno ) ) goto 03
      number = numb

      nn = 0

0505  nn = nn + 1
 05   write(*,06) nn
 06   format(//' Fuel component number'i3' properties list'/
     +         ' Name [ 12 characters or fewer ]   ',$)
      read(*,08) name1
      parts( nn ) = name1
 08   format(a12)

 09   write(*,10) name1
 10   format(1x,a12' dry loading, kg / sq m    ',$)
      read(*,*,err=09) drywt
      call retry( drywt , small , big , test )
      if( test ) goto 09
      wdry( nn ) = drywt

 11   write(*,12) name1
 12   format(1x,a12' total mineral ash fraction   ',$)
      read(*,*,err=11) ashes
      call retry( ashes , ash1 , ash2 , test )
      if( test ) goto 11
      ash( nn ) = ashes

 13   write(*,14) name1
```

```
14  format(1x,a12' heat of combustion, J / kg    ',$)
    read(*,*,err=13) hots
    call retry( hots , htv1 , htv2 , test )
    if( test ) goto 13
    htval( nn ) = hots

15  write(*,16) name1
16  format(1x,a12' moisture content, fraction    ',$)
    read(*,*,err=15) fms
    call retry( fms , fms1 , fms2 , test )
    if( test ) goto 15
    fmois( nn ) = fms

17  write(*,18) name1
18  format(1x,a12' ovendry mass density, kg / cubic m    ',$)
    read(*,*,err=17) dryd
    call retry( dryd , den1 , den2 , test )
    if( test ) goto 17
    dendry( nn ) = dryd

19  write(*,20) name1
20  format(1x,a12' surface / volume ratio , inverse m    ',$)
    read(*,*,err=19) sigs
    call retry( sigs , sig1 , sig2 , test )
    if( test ) goto 19
    sigma( nn ) = sigs

21  write(*,22) name1
22  format(1x,a12' specific heat capacity ovendry , J / kg deg K  ',$)
    read(*,*,err=21) cpd
    call retry( cpd , cht1 , cht2 , test )
    if( test ) goto 21
    cheat( nn ) = cpd

23  write(*,24) name1
24  format(1x,a12' ovendry thermal conductivity , W / m deg K    ',$)
    read(*,*,err=23) cond
    call retry( cond , con1 , con2 , test )
    if( test ) goto 23
    condry( nn ) = cond

25  write(*,26) name1
26  format(1x,a12' ignition temperature, deg C    ',$)
    read(*,*,err=25) tigi
    call retry( tigi , tig1 , tig2 , test )
    if( test ) goto 25
    tpig( nn ) = tigi + 273.0

27  write(*,28) name1
28  format(1x,a12' char [end pyrolysis] temperature, deg C    ',$)
    read(*,*,err=27) tchi
    call retry( tchi , tch1 , tch2 , test )
    if( test ) goto 27
    tchar( nn ) = tchi + 273.0
```

```
      if( nn .LT. number ) goto 0505
      if( ( next .EQ. 1 ) .AND. ( ido .EQ. 1 ) ) goto 50

 29   write(*,30)
 30   format(' Enter igniting surface fire intensity , kW / sq m   ',$)
      read(*,*,err=29) fir
      call retry( fir , fir1 , fir2 , test )
      if( test ) goto 29
      fi = fir

 31   write(*,32)
 32   format(' Enter igniting surface fire residence time , s    ',$)
      read(*,*,err=31) tir
      call retry( tir , ti1 , ti2 , test )
      if( test ) goto 31
      ti = tir

 33   write(*,34)
 34   format(' Windspeed at top of fuelbed , m / s    ',$)
      read(*,*,err=33) uin
      call retry( uin , u1 , u2 , test )
      if( test ) goto 33
      u = uin

 35   write(*,36)
 36   format(' Depth of fuelbed , m    ',$)
      read(*,*,err=35) din
      call retry( din , d1 , d2 , test )
      if( test ) goto 35
      d = din

 37   write(*,38)
 38   format(' Ambient temperature , deg C    ',$)
      read(*,*,err=37) tamb
      call retry( tamb , tam1 , tam2 , test )
      if( test ) goto 37
      tpamb = tamb + 273.0

 39   write(*,40)
 40   format(' Dimensionless area influence factor [ak parameter]    ',$)
      read(*,*,err=39) ak

 41   write(*,42)
 42   format(' Fire environment minimum temperature parameter r0    ',$)
      read(*,*,err=41) r0

 43   write(*,44)
 44   format(' Fire environment increment temperature parameter dr   ',$)
      read(*,*,err=43) dr

 45   write(*,46)
 46   format(' Time step for integration of burning rates , s    ',$)
      read(*,*,err=45) dt

 47   write(*,48)
```

```
 48   format(' Number time steps [ must exceed 1 ]   ',$)
      read(*,*,err=47) ntimes
      if( ntimes .LE. 1 ) goto 47

 49   write(*,491)
 491  format(' Duff dry weight loading, kg / sq m   ',$)
      read(*,*,err=49) wdf
      call retry( wdf , wdf1 , wdf2 , test )
      if( test ) goto 49

 492  write(*,493)
 493  format(' Duff moisture fraction    ',$)
      read(*,*,err=492) dfm
      call retry( dfm , dfm1 , dfm2 , test )
      if( test ) goto 492

c  All data collected --- now time to review and revise as needed

 50   write(*,52)
 52   format(' All data can be reviewed and revised now.  Enter:'/
     +       ' 0 to archive and/or execute present data'/
     +       ' 1 to review fuel component properties'/
     +       ' 2 to review igniting fire and environmental data'/
     +       ' 3 to review internal and control variables'/
     +       ' 4 to terminate program now    ',$)
      read(*,*,err=50) next
      if( next .EQ. 4 ) stop' Terminated'
      if( next .EQ. 0 ) goto 2000
      if( ( next .LT. 1 ) .OR. ( next .GT. 3 ) ) goto 50
      if( next .EQ. 1 ) then

c  Review fuel component properties

 53     write(*,54)
 54     format(' Enter 0 to terminate review & revision of fuel data'/
     +         ' Enter 1 to select a fuel component for review'/
     +         ' Enter 2 to add or delete a fuel component    ',$)
        read(*,*,err=53) ido
        if( ido .EQ. 0 ) goto 50
        if( ido .EQ. 2 ) goto 66
        if( ido .NE. 1 ) goto 53
 55     write(*,56) ( n ,  parts( n ) , wdry( n ) , n = 1 , number )
 56     format(i3,3x,a12,'  loading = 'e12.3'   kg  / sq m')
 57     write(*,58)
 58     format(' Number component to be revised [ 0 = prev menu ]  ',$)
        read(*,*,err=57) nn
        if( nn .EQ. 0 )goto 53
        if( ( nn .LT. 1 ) .OR. ( nn .GT. number ) ) goto 55
 59     write(*,60) parts( nn ) , wdry( nn ) , ash( nn ) ,
     +          htval( nn ) , fmois( nn ) , dendry( nn ) ,
     +          sigma( nn ) , cheat( nn ) , condry( nn ) ,
     +        ( tpig( nn ) - 273.0 ) , ( tchar( nn ) - 273.0 )
 60     format(' index     parameter  data entered'/
     +         '   1'10x'name'4x,a12/
     +         '   2'7x'loading  ',e12.3/
```

```
     +          '   3'7x'ash frac ',e12.3/
     +          '   4'7x'heat comb',e12.3/
     +          '   5'7x'mois frac',e12.3/
     +          '   6'7x'density  ',e12.3/
     +          '   7'7x'surf/vol ',e12.3/
     +          '   8'7x'heat capy',e12.3/
     +          '   9'7x'conduct y',e12.3/
     +          '  10'7x'ig temp C',e12.3/
     +          '  11'7x'char temp',e12.3/
     +          '  enter number param to change [ 0 = prev menu ]  ',$)
          read(*,*,err=59) idn
          if( idn .EQ. 0 ) goto 55
          if( idn .GT. 11 ) goto 59
          if( idn .EQ. 1 ) then
             write(*,*)' Enter new name - 12 char or fewer'
             read(*,0 8)parts( nn )
             goto 59
          end if
 61       write(*,62)
 62       format(' New value = ?    ',$)
          read(*,*,err=61) value
          if( idn .EQ. 2 ) wdry( nn ) = value
          if( idn .EQ. 3 ) ash( nn ) = value
          if( idn .EQ. 4 ) htval( nn ) = value
          if( idn .EQ. 5 ) fmois( nn ) = value
          if( idn .EQ. 6 ) dendry( nn ) = value
          if( idn .EQ. 7 ) sigma( nn ) = value
          if( idn .EQ. 8 ) cheat( nn ) = value
          if( idn .EQ. 9 ) condry( nn ) = value
          if( idn .EQ. 10 ) tpig( nn ) = value + 273.0
          if( idn .EQ. 11 ) tchar( nn ) = value + 273.0
          goto 59

c  Add or delete a fuel component

 66       write(*,68)
 68       format(' Enter - 1 to delete a fuel component'/
     +          ' Enter 0 if no more additions or deletions'/
     +          ' Enter + 1 to add a fuel component   ',$)
          read(*,*,err=66) ido
          if( ido .EQ. 0 ) goto 50
          if( ido .EQ. -1 ) then

c  Delete a fuel component

 69          write(*,56) ( n , parts( n ) , wdry( n ) , n = 1 , number )
             write(*,70)
 70          format(' Index number of component to delete   ',$)
             read(*,*,err=69) ind
             if( ( ind .LE. 0 ) .OR. ( ind .GT. number ) ) goto 66
             if( ind .EQ. number ) then
                number = number - 1
                goto 66
             end if
             parts( ind ) = parts( number )
```

```
          wdry( ind ) = wdry( number )
          ash( ind ) = ash( number )
          htval( ind ) = htval( number )
          fmois( ind ) = fmois( number )
          dendry( ind ) = dendry( number )
          sigma( ind ) = sigma( number )
          cheat( ind ) = cheat( number )
          condry( ind ) = condry( number )
          tpig( ind ) = tpig( number )
          tchar( ind ) = tchar( number )
          number = number - 1
          goto 66
        end if

c  Add a fuel component

        if( ido .NE. 1 ) goto 66
        nn = number
        number = number + 1
        goto 0505
      end if

      if( next .EQ. 2 ) then

c  Review igniting fire and environmental data

 71     write(*,72) fi
 72     format('  1 - Igniting fire intensity, kW / sq m          'e12.3)
        write(*,73) ti
 73     format('  2 - Igniting fire residence time, s             'e12.3)
        write(*,74) u
 74     format('  3 - windspeed at top of fuel bed, m / s        'e12.3)
        write(*,75) d
 75     format('  4 - fuelbed depth, m                           'e12.3)
        write(*,76) ( tpamb - 273.0 )
 76     format('  5 - ambient temperature, deg C                 'e12.3)
 77     write(*,78)
 78     format(' Index of parameter to change [ 0 = prev menu ]    ',$)
        read(*,*,err=77) ind
        if( ind .EQ. 0 ) goto 50
        if( ( ind .LT. 1 ) .OR. ( ind .GT. 5 ) ) goto 71
        write(*,62)
        read(*,*,err=71) value
        if( ind .EQ. 1 ) fi = value
        if( ind .EQ. 2 ) ti = value
        if( ind .EQ. 3 ) u = value
        if( ind .EQ. 4 ) d = value
        if( ind .EQ. 5 ) tpamb = value + 273.0
        goto 71
      end if

c  input quantity  next .EQ. 3  by fallthrough to this point, so
c  Review internal and control variables

 79   write(*,80) ak
```

```
      write(*,81) r0
      write(*,82) dr
      write(*,83) dt
      write(*,84) wdf
      write(*,85) dfm
      write(*,86) ntimes
      write(*,87)
 80   format('  1 - Area influence factor [ak parameter]          'e12.3)
 81   format('  2 - Fire environment temperature parameter r0     'e12.3)
 82   format('  3 - Fire environment temperature parameter dr     'e12.3)
 83   format('  4 - Time step to integrate burning rates , s      'e12.3)
 84   format('  5 - Duff dry weight loading , kg / sq m           'e12.3)
 85   format('  6 - Duff moisture content , fraction dry weight   'e12.3)
 86   format('  7 - Number time steps                             'i4)
 87   format('  Index of parameter to change [ 0 = prev menu ]    ',$)
      read(*,*,err=79) ind
      if( ind .EQ. 0 ) goto 50
      if( ( ind .LT. 1 ) .OR. ( ind .GT. 7 ) ) goto 79
      write(*,62)
      if( ind .LT. 7 ) read(*,*,err=79) value
      if( ind .EQ. 1 ) ak = value
      if( ind .EQ. 2 ) r0 = value
      if( ind .EQ. 3 ) dr = value
      if( ind .EQ. 4 ) dt = value
      if( ind .EQ. 5 ) wdf = value
      if( ind .EQ. 6 ) dfm = value
      if( ind .EQ. 7 ) read(*,*,err=79) ntimes
      goto 79

C End of keyboard entry and review-data / revise-data segment
C Following are file retrieval [ 1000 ] and archiving [ 2000 ] segments

1000  write(*,1001)
1001  format(' Enter name of file holding fuel component data  ',$)
      read(*,08,err=1000) infile
1002  open(77,file=infile,status='OLD',form='FORMATTED',err=1022)
      read(77,1003) number
1003  format(i6)
      do n = 1 , number
         read(77,2004) parts( n )
         read(77,2005) wdry( n ) , ash( n ) , htval( n ) , fmois( n ) ,
     +                 dendry( n ) , sigma( n ) , cheat( n ) ,
     +                 condry( n ) , tpig( n ) , tchar( n )
      end do
      close(77)
      write(*,1004)
1004  format(' Enter name of file holding igniting fire, environmental'/
     +       ', and program control parameters   ',$)
      read(*,08) infile
1005  open(77,file=infile,status='OLD',form='FORMATTED',err=1025)
      read(77,2008) fi , ti , u , d , tpamb
      read(77,2009) ak , r0 , dr , dt , ntimes
      read(77,2010,end=1006) wdf , dfm
      close(77)
      goto 50
```

```
 1006 wdf = 0.
      dfm = 0.
      close(77)
      goto 50
 1022 write(*,1030) infile
 1030 format(' Error opening input file  ',a12'  reenter file name')
      read(*,08,err=1022) infile
      goto 1002
 1025 write(*,1030) infile
      read(*,08,err=1025) infile
      goto 1005

c  Archive current run conditions?

 2000 write(*,2100)
 2100 format(' Enter 2 for previous menu'/
     +       ' Enter 1 to archive current data set'/
     +       ' Enter 0 to execute without archiving     ',$)
      read(*,*,err=2000) ido
      if( ido .EQ. 2 ) goto 50
      if( ido .EQ. 0 ) return
      if( ido .NE. 1 ) goto 2000
 2001 write(*,2002)
 2002 format(' Enter file name [ 12 char or fewer ] for storage of the'/
     +       ' fuel component data currently in use   ',$)
      read(*,08,err=2001) outfil
 2022 open(66,file=outfil,status='UNKNOWN',form='FORMATTED',err=2021)
      write(66,2003) number
 2003 format(i6,' Fuel component properties -- 3 records per entry')
      do n = 1 , number
         write(66,2004) parts( n )
         write(66,2005) wdry( n ) , ash( n ) , htval( n ) , fmois( n ) ,
     +                  dendry( n ) , sigma( n ) , cheat( n ) ,
     +                  condry( n ) , tpig( n ) , tchar( n )
 2004    format(2x,a12)
 2005    format(2x,5e15.3/2x,5e15.3)
      end do
      close(66)
 2006 write(*,2007)
 2007 format(' Enter file name [ 12 char or fewer ] for storage of the'/
     +       ' igniting fire, environmental, and control data used  ',$)
      read(*,08,err=2006) outfil
 2027 open(66,file=outfil,status='UNKNOWN',form='FORMATTED',err=2026)
      write(66,2008) fi , ti , u , d , tpamb
 2008 format(2x,5e15.3)
      write(66,2009) ak , r0 , dr , dt , ntimes
 2009 format(2x,4e15.3,i6)
      write(66,2010) wdf , dfm
 2010 format(2x,2e15.3)
      close(66)
      goto 2000
 2021 write(*,2030) outfil
 2030 format(' Error opening output file  'a12'  enter another name')
      goto 2001
 2026 write(*,2030) outfil
```

```
      read(*,08,err=2026) outfil
      goto 2027

      end



      subroutine RETRY( value , valo , vahi , test )
      logical test

      test = ( value .GT. valo ) .AND. ( value .LT. vahi )
      if( test ) then
         test = .FALSE.
         return
      end if

      test = ( value .LE. valo )
      if( test ) then
01       write(*,*)' Value entered seems small'
         write(*,*)' Enter 0 to proceed with small value'
         write(*,*)' Enter 1 to provide a new value'
         read(*,*,err=01 )in
         if( in .EQ. 0 ) then
            test = .FALSE.
            return
         end if
         if( in .NE. 1 ) goto 01
         return
      end if

      test = ( value .GE. vahi )
      if( test ) then
02       write(*,*)' Value entered seems large'
         write(*,*)' Enter 0 to proceed with large value'
         write(*,*)' Enter 1 to provide a new value'
         read(*,*,err=02) in
         if( in .EQ. 0 ) then
            test = .FALSE.
            return
         end if
         if( in .NE. 1 ) goto 02
         return
      end if
      end
```

```
 subroutine ARRAYS( maxno , number , wdry , ash , dendry , fmois ,
+                   sigma , htval , cheat , condry , tpig , tchar ,
+                   diam , key , work , ak , elam , alone , xmat ,
+                   wo , maxkl , parts , list , area )
```

Orders the fuel description arrays according to the paradigm described in subroutine SORTER and computes the interaction matrix xmat from the array elam and the list alone returned from subroutine OVLAPS. Parameters in arrays are defined in terms of initial values as:

wdry	ovendry mass loading , kg / sq m
ash	mineral content , fraction dry mass
dendry	ovendry mass density , kg / cu m
fmois	moisture content , fraction dry mass
sigma	surface to volume ratio , 1 / m
htval	low heat of combustion , J / kg
cheat	specific heat capacity , (J / K) / kg dry mass
condry	thermal conductivity , W / m K , ovendry
tpig	ignition temperature , K
tchar	char temperature , K
diam	initial diameter , m [by interaction pairs]
key	ordered index list
work	workspace array
elam	interaction matrix from OVLAPS
alone	noninteraction fraction list from OVLAPS
xmat	consolidated interaction matrix
wo	initial dry loading by interaction pairs
area	fraction of site area expected to be covered at least once by initial planform area of ea size

```
  character*12 parts( maxno ) , list( maxno )
  real*4 wdry( maxno ) , ash( maxno ) , dendry( maxno )
  real*4 fmois( maxno ) , sigma( maxno ) , htval( maxno )
  real*4 cheat( maxno ) , condry( maxno ) , tpig( maxno )
  real*4 tchar( maxno ) , work( maxno )
  real*4 elam( maxno , maxno ) , alone( maxno )
  real*4 area( maxno )
  real*4 xmat( maxkl ) , diam( maxkl ) , wo( maxkl )

  integer key( maxno )

  loc( k , l ) = k * ( k + 1 ) / 2 + l

  call SORTER( maxno , number , sigma , fmois , dendry , key )

  do j = 1 , number
      k = key( j )
      list( j ) = parts( k )
  end do
  do j = 1 , number
      parts( j ) = list( j )
  end do

  do j = 1 , number
```

```
      k = key( j )
      work( j ) = wdry( k )
end do
do j = 1 , number
      wdry( j ) = work( j )
end do

do j = 1 , number
      k = key( j )
      work( j ) = ash( k )
end do
do j = 1 , number
      ash( j ) = work( j )
end do

do j = 1 , number
      k = key( j )
      work( j ) = htval( k )
end do
do j = 1 , number
      htval( j ) = work( j )
end do

do j = 1 , number
      k = key( j )
      work( j ) = cheat( k )
end do
do j = 1 , number
      cheat( j ) = work( j )
end do

do j = 1 , number
      k = key( j )
      work( j ) = condry( k )
end do
do j = 1 , number
      condry( j ) = work( j )
end do

do j = 1 , number
      k = key( j )
      work( j ) = tpig( k )
end do
do j = 1 , number
      tpig( j ) = work( j )
end do

do j = 1 , number
      k = key( j )
      work( j ) = tchar( k )
end do
do j = 1 , number
      tchar( j ) = work( j )
end do
```

```
 call OVLAPS( wdry , sigma , dendry , ak , number , maxno , maxkl ,
+                                     xmat , elam , alone , area )

 do k = 1 , number
    diak = 4. / sigma( k )
    wtk = wdry( k )
    kl = loc( k , 0 )
    diam( kl ) = diak
    xmat( kl ) = alone( k )
    wo( kl ) = wtk * xmat( kl )
    do j = 1 , k
       kj = loc( k , j )
       diam( kj ) = diak
       xmat( kj ) = elam( k , j )
       wo( kj ) = wtk * xmat( kj )
    end do
 end do

 return
 end
```

```
      subroutine SORTER( maxno , number , sigma , fmois , dryden , key )

Sorts fuel element list in order of increasing size (decreasing sigma)
For elements with same size, order determined on increasing moisture
content (fmois).  If items have same size and moisture content, order
on the basis of increasing mass density (dryden).  "number" elements are
included in the list, which has a maximum length of "maxno".  The integer
list:  key( j ) , j = 1 , number holds the indices in order, so other
fuel parameters can be ordered and associated as necessary.

      real*4 sigma( maxno ) , fmois( maxno ) , dryden( maxno )
      integer key( maxno )
      logical diam , mois , dens , tied

      do j = 1 , maxno
         key( j ) = j
      end do

Replacement sort: order on increasing size , moisture , density

      do j = 2 , number
         s = 1. / sigma( j )
         fm = fmois( j )
         de = dryden( j )
         keep = key( j )
         do i = ( j - 1 ) , 1 , -1
            usi = 1. / sigma( i )
            diam = ( usi .LT. s )
            if( diam ) goto 10
            tied = ( usi .EQ. s )
            if( .NOT. tied ) goto 05
            mois = ( fmois( i ) .LT. fm )
            if( mois ) goto 10
            tied = ( fmois( i ) .EQ. fm )
            if( .NOT. tied ) goto 05
            dens = ( dryden( i ) .LE. de )
            if( dens ) goto 10
05          sigma( i + 1 ) = sigma( i )
            fmois( i + 1 ) = fmois( i )
            dryden( i + 1 ) = dryden( i )
            key( i + 1 ) = key( i )
         end do
         i = 0
10       sigma( i + 1 ) = 1. / s
         fmois( i + 1 ) = fm
         dryden( i + 1 ) = de
         key( i + 1 ) = keep
      end do

      return
      end
```

```
   subroutine OVLAPS( dryld , sigma , dryden , ak , number , maxno ,
  +                                  maxkl , beta , elam , alone , area )
```

Computes the interaction matrix elam(j , k) which apportions the influence of smaller and equal size pieces on each size class for the purpose of establishing the rates at which the elements burn out. Input quantities are: dryld , the ovendry mass per unit area of each element available for burning after the passage of the igniting surface fire; sigma , the particle's surface / volume ratio , and dryden , the ovendry mass density of the particle; ak a dimensionless parameter that scales the planform area of a particle to its area of influence. There are "number" separate particle classes, of a maximum number = maxno. It is assumed that the lists are ordered on size class (nonincreasing surface / volume ratio). List "alone" gives the fraction of each loading that is not influenced by any other category.

```
real*4 dryld( maxno ) , sigma( maxno ) , dryden( maxno )
real*4 beta( maxkl ) , elam( maxno , maxno ) , alone( maxno )
real*4 area( maxno )

loc( k , l ) = k * ( k + 1 ) / 2 + l

pi = abs( acos( -1. ) )
do j = 1 , number
   alone( j ) = 0.
   do k = 1 , j
      kj = loc( j , k )
      beta( kj ) = 0.
   end do
   do k = 1 , number
      elam( j , k ) = 0.
   end do
end do

do k = 1 , number
   siga = ak * sigma( k ) / pi
   do l = 1 , k
      kl = loc( k , l )
      a = siga * dryld( l ) / dryden( l )
      if( k .EQ. l ) then
         bb = 1. - exp( - a )
         area( k ) = bb
      end if
      if( k .NE. l ) bb = min( 1. , a )
      beta( kl ) = bb
   end do
end do

if( number . EQ. 1 ) then
   elam( 1 , 1 ) = beta( 2 )
   alone( 1 ) = 1. - elam( 1 , 1 )
   return
end if

do k = 1 , number
```

```
      frac = 0.
      do l = 1 , k
         kl = loc( k , l )
         frac = frac + beta( kl )
      end do
      if( frac .GT. 1. ) then
         do l = 1 , k
            kl = loc( k , l )
            elam( k , l ) = beta( kl ) / frac
         end do
         alone( k ) = 0.
      else
         do l = 1 , k
            kl = loc( k , l )
            elam( k , l ) = beta( kl )
         end do
         alone( k ) = 1. - frac
      end if
   end do

   return
   end
```

```
   subroutine START( dt , mxstep , now , maxno , number , wo , alfa ,
  +                  dendry , fmois , cheat , condry , diam , tpig ,
  +                  tchar , xmat , tpamb , tpdry , fi , flit , fout ,
  +                  tdry , tign , tout , qcum , tcum , acum , qdot ,
  +                  ddot , wodot , work , u , d , r0 , dr , ch2o ,
  +                  ncalls , maxkl )
```

This routine initializes variables prior to starting sequence of calls to subroutine STEP. On input here, fi is area intensity of spreading fire , dt is the residence time for the spreading fire. Only physical parameters specified are the fuel array descriptors. To call STEP , one must initialize the following variables.

Input parameters:

dt =	spreading fire residence time , sec
mxstep =	max dimension of historical sequences
now =	index marks end of time step
maxno =	max number of fuel components
number =	actual number of fuel components
wo =	current ovendry loading for the larger of each component pair, kg / sq m
alfa =	dry thermal diffusivity of component , sq m / s
dendry =	ovendry density of component , kg / cu m
fmois =	moisture fraction of component
cheat =	specific heat capacity of component , J / kg K
condry =	ovendry thermal conductivity , W / sq m K
diam =	current diameter of the larger of each fuel component pair , m
tpig =	ignition temperature (K) , by component
tchar =	end - pyrolysis temperature (K) , by component
xmat =	table of influence fractions between components
tpamb =	ambient temperature (K)
fi =	current fire intensity (site avg) , kW / sq m

Parameters updated [input and output]

ncalls =	counter of calls to this routine ... = 0 on first call or reset ; cumulates after first call
flit =	fraction of each component currently alight
fout =	fraction of each component currently gone out
tdry =	time of drying start of the larger of each fuel component pair
tign =	ignition time for the larger of each fuel component pair
tout =	burnout time of larger component of pairs
qcum =	cumulative heat input to larger of pair , J / sq m
tcum =	cumulative temp integral for qcum (drying)
acum =	heat pulse area for historical rate averaging
qdot =	history (post ignite) of heat transfer rate to the larger of each component pair
ddot =	diameter reduction rate , larger of pair , m / s
wodot =	dry loading loss rate for larger of pair

```
Constant parameters

     u =              mean horizontal windspeed at top of fuelbed
     d =              fuelbed depth
     r0 =             minimum value of mixing parameter
     dr =             max - min value of mixing parameter
     ch2o =           specific heat capacity of water , J / kg K
     hvap =           heat of vaporization of water J / kg
     tpdry =          temperature ( all components ) start drying ( K )

   real*4 wo( maxkl ) , alfa( maxno ) , dendry( maxno )
   real*4 fmois( maxno ) , cheat( maxno )
   real*4 condry( maxno ) , diam( maxkl ) , tpig( maxno )
   real*4 tchar( maxno ) , xmat( maxkl )
   real*4 tdry( maxkl ) , tign( maxkl )
   real*4 tout( maxkl ) , qcum( maxkl )
   real*4 tcum( maxkl ) , acum( maxkl )
   real*4 qdot( maxkl , mxstep ) , ddot( maxkl )
   real*4 wodot( maxkl ) , flit( maxno ) , fout( maxno )
   real*4 work( maxno )

   loc( k , l ) = k * ( k + 1 ) / 2 + l

   data rindef / 1.e+30 /

   ch2o = 4186.
   tpdry = 353.

Initialize time varying quantities and set up work( k )
The diameter reduction rate of fuel component k is given
by the product of the rate of heat transfer to it, per
unit surface area, and the quantity work( k )

   do k = 1 , number
      fout( k ) = 0.
      flit( k ) = 0.
      alfa( k ) = condry( k ) / ( dendry( k ) * cheat( k ) )
      effect of moisture content on burning rate (scale factor)
      delm = 1.67* fmois( k )
      effect of component mass density (empirical)
      heatk = dendry( k ) / 446.
      empirical burn rate factor, J / cu m - K
      heatk = heatk * 2.01e+06 * ( 1. + delm )
      normalize out driving temperature difference (Tfire - Tchar)
      to average value of lab experiments used to find above constants
      work( k ) = 1. / ( 255. * heatk )
      do l = 0 , k
         kl = loc( k , l )
         tout( kl ) = rindef
         tign( kl ) = rindef
         tdry( kl ) = rindef
         tcum( kl ) = 0.
         qcum( kl ) = 0.
      end do
   end do
```

```
Make first estimate of drying start times for all components
These times are usually brief and make little or no difference

   r = r0 + 0.25 * dr
   tf = tempf( fi , r , tpamb )
   ts = tpamb
   if( tf .LE. ( tpdry + 10. ) ) stop' Igniting fire cannot dry fuel'
   thd = ( tpdry - ts ) / ( tf - ts )
   tx = 0.5 * ( ts + tpdry )

   do k = 1 , number
      factor = dendry( k ) * fmois( k )
      conwet = condry( k ) + 4.27e-04 * factor
      do l = 0 , k
         kl = loc( k , l )
         dia = diam( kl )
         call heatx( u , d , dia , tf , tx , hf , hb , conwet , en )
         call DRYTIM( en , thd , dryt )
         cpwet = cheat( k ) + fmois( k ) * ch2o
         fac = ( ( 0.5 * dia ) ** 2 ) / conwet
         fac = fac * dendry( k ) * cpwet
         dryt = fac * dryt
         tdry( kl ) = dryt
      end do
   end do

Next , determine which components are alight in spreading fire

   tsd = tpdry

   do k = 1 , number
      c = condry( k )
      tigk = tpig( k )
      do l = 0 , k
         kl = loc( k , l )
         dryt = tdry( kl )
         if( dryt .GE. dt ) goto 10
         dia = diam( kl )
         ts = 0.5*( tsd + tigk )
         call heatx( u , d , dia , tf , ts , hf , hb , c , e )
         tcum( kl ) = max( ( tf - ts ) * ( dt - dryt ) , 0. )
         qcum( kl ) = hb * tcum( kl )
         if( tf .LE. ( tigk + 10. ) ) goto 10
         call TIGNIT( tpamb , tpdry , tpig( k ) , tf , condry( k ),
     +                cheat( k ) , fmois( k ) , dendry( k ) , hb ,
     +                dtign )
         trt = dryt + dtign
         tign( kl ) = 0.5 * trt
         if( dt .GT. trt ) flit( k ) = flit( k ) + xmat( kl )
10       continue
      end do
   end do

   nlit = 0
```

```
   trt = rindef

Determine minimum ignition time and verify ignition exists

   do k = 1 , number
      if( flit( k ) .GT. 0. ) nlit = nlit + 1
      do l = 0 , k
         kl = loc( k , l )
         trt = min( trt , tign( kl ) )
      end do
   end do

   if( nlit .EQ. 0 ) stop' START ignites no fuel'

Deduct trt from all time estimates , resetting time origin

   do k = 1 , number
      do l = 0 , k
         kl = loc( k , l )
         if( tdry( kl ) .LT. rindef ) then
            tdry( kl ) = tdry ( kl ) - trt
         end if
         if( tign( kl ) .LT. rindef ) then
            tign( kl ) = tign( kl ) - trt
         end if
      end do
   end do

Now go through all component pairs and establish burning rates
for all the components that are ignited; extrapolate to end time dt

   do k = 1 , number
      if( flit( k ) .EQ. 0. ) then
         do l = 0 , k
            kl = loc( k , l )
            ddot( kl ) = 0.
            tout( kl ) = rindef
            wodot( kl ) = 0.
         end do
      else
         ts = tchar( k )
         c = condry( k )
         do l = 0 , k
            kl = loc( k , l )
            dia = diam( kl )
            call heatx( u , d , dia , tf , ts , hf , hb , c , e )
            qdot( kl , now ) = hb * max( ( tf - ts ) , 0. )
            aint = ( c / hb ) ** 2
            ddt = dt - tign( kl )
            acum( kl ) = aint * ddt
            ddot( kl ) = qdot( kl , now ) * work( k )
            tout( kl ) = dia / ddot( kl )
            dnext = max( 0. , ( dia - ddt * ddot( kl ) ) )
            wnext = wo( kl ) * ( ( dnext / dia ) ** 2 )
            wodot( kl ) = ( wo( kl ) - wnext ) / ddt
```

```
            diam( kl ) = dnext
            wo( kl ) = wnext
            df = 0.
            if( dnext .LE. 0. ) then
               df = xmat( kl )
               wodot( kl ) = 0.
               ddot( kl ) = 0.
            end if
            flit( k ) = flit( k ) - df
            fout( k ) = fout( k ) + df
         end do
      end if
   end do


   ncalls = 0

   return
   end
```

```
      subroutine FIRINT( wodot , ash , htval , maxno , number , maxkl ,
     +                   area , fint , fi )

Computes fi = site avg fire intensity given the burning rates of all
interacting pairs of fuel components [ wodot ] , the mineral ash content
of each component [ ash ] , the heat of combustion value [ htval ] for
each , and the number of fuel components [ number ] , where max = maxno.
fi is in kW / sq m , while htval is in J / kg.

fint( k ) is the correction to fi to adjust
the intensity level to be the local value where size k is burning.

      real*4 ash( maxno ) , htval( maxno )
      real*4 wodot( maxkl ) , area( maxno ) , fint( maxno )
      data small / 1.e-06 /

      loc( k , l ) = k * ( k + 1 ) / 2 + l

      sum = 0.
      do k = 1 , number
         wdotk = 0.
         do l = 0 , k
            kl = loc( k , l )
            wdotk = wdotk + wodot( kl )
         end do
         term = ( 1. - ash( k ) ) * htval( k ) * wdotk * 1.e-03
         ark = area( k )
         if( ark .GT. small ) then
            fint( k ) = term / ark - term
         else
            fint( k ) = 0.
         end if
         sum = sum + term
      end do

      fi = sum
      return
      end
```

```
      subroutine STASH( time , now , maxno , number , outfil , fi ,
     +                  flit , fout , wo , wodot , diam , ddot ,
     +                  tdry , tign , tout , fmois , maxkl , nun )
```

This routine stashes output from the BURNUP model package on a snapshot basis. Every time it is called, it "dumps" a picture of the status of each component of the fuel complex, as a table of interacting pairs.

```
      real*4 wo( maxkl ) , wodot( maxkl )
      real*4 diam( maxkl ) , ddot( maxkl )
      real*4 flit( maxno ) , fout( maxno )
      real*4 tdry( maxkl ) , tign( maxkl )
      real*4 tout( maxkl ) , fmois( maxno )
      character*12 outfil , histry
      logical snaps

      loc( k , l ) = k * ( k + 1 ) / 2 + l

      if( now .EQ. 1 ) then
         wd = 0.
         wg = 0.
         do m = 1 , number
            fmm = fmois( m )
            wdm = 0.
            do n = 0 , m
               mn = loc( m , n )
               wdm = wdm + wo( mn )
            end do
            wgm = wdm * ( 1. + fmm )
            wd = wd + wdm
            wg = wg + wgm
         end do
         wd0 = wd
         wg0 = wg
         snaps = .FALSE.
         nun = 77
         mum = 66
         histry = 'HISTORY.DAT'
         open( mum , file=histry , status='UNKNOWN' , form='FORMATTED' )
1000     write( * , 1001 )
1001     format(' Enter 1 to stash fuelbed snapshots, 0 to skip   ',$)
         read( * , * ) istash
         if( istash .EQ. 0 ) goto 06
         if( istash .NE. 1 ) goto 1000
         snaps = .TRUE.
01       write( * , 02 )
02       format(' File name [ max 12 char ] for STASH output   ',$)
         read(*,03) outfil
03       format(a12)
         open(nun,file=outfil,status='NEW',form='FORMATTED',err=04)
         goto 06
04       write(*,05) outfil
05       format(' Error opening file  'a12/
     +          ' Enter 0 to try a new name'/
     +          ' Enter 1 to overwrite existing file'/
```

```
     +          ' Enter 2 to terminate program now   ',$)
          read(*,*) ido
          if( ido .EQ. 2 ) stop' Program ended'
          if( ido .EQ. 0 ) goto 01
          if( ido .NE. 1 ) goto 04
          open(nun,file=outfil,status='UNKNOWN',form='FORMATTED',err=04)
        end if

06      continue

        if( snaps ) write( nun , 07 ) time , fi
07      format( 8e10.3 )

        wd = 0.
        wg = 0.
        do m = 1 , number
          fmm = fmois( m )
          wdm = 0.
          if( snaps ) write( nun , 07 ) time , flit( m ) , fout( m )
          do n = 0 , m
            mn = loc( m , n )
            wdm = wdm + wo( mn )
            if( snaps ) write( nun , 07 ) time , wo( mn ) ,
     +                  wodot( mn ) , diam( mn ) , ddot( mn ) ,
     +                   tdry( mn ) , tign( mn ) , tout( mn )
          end do
          wgm = wdm * ( 1. + fmm )
          wd = wd + wdm
          wg = wg + wgm
        end do
        wgf = wg / wg0
        wdf = wd / wd0

        write( mum , 07 ) time , wg , wd , wgf , wdf , fi

        return
        end
```

```
      subroutine TIGNIT( tpam , tpdr , tpig , tpfi , cond ,
     +                   chtd , fmof , dend , hbar , tmig )

  tpam = ambient temperature , K
  tpdr = fuel temperature at start of drying , K
  tpig = fuel surface temperature at ignition , K
  tpfi = fire environment temperature , K
  cond = fuel ovendry thermal conductivity , W / m K
  chtd = fuel ovendry specific heat capacity , J / kg K
  fmof = fuel moisture content , fraction dry weight
  dend = fuel ovendry density , kg / cu m
  hbar = effective film heat transfer coefficient [< HEATX] W / sq m K
  tmig = predicted time to piloted ignition , s

      data a03 / -1.3371565 / , a13 / 0.4653628 / , a23 / -0.1282064 /
      data pinv / 2.125534 / , small / 1.e-06 /
      data hvap / 2.177e+06 / , cpm / 4186. / , conc / 4.27e-04 /

  approximate function of beta to be solved is ff( x ) where
  x  =  1 / ( 1 + p * beta )  { Hastings, Approximations for
  digital computers } and we employ      pinv  =  1 / p

      ff( x ) = b03 + x * ( a13 + x * ( a23 + x ) )

  radiant heating equivalent form gives end condition fixes beta

      b03 = a03 * ( tpfi - tpig ) / ( tpfi - tpam )

  find x that solves ff( x ) = 0 ;  method is binary search

      xlo = 0.
      xhi = 1.
01    xav = 0.5 * ( xlo + xhi )
      fav = ff( xav )
      if( abs( fav ) .LE. small ) goto 10
      if( fav .LT. 0. ) xlo = xav
      if( fav .GT. 0. ) xhi = xav
      goto 01

10    beta = pinv * ( 1. - xav ) / xav
      conw = cond + conc * dend * fmof
      dtb = tpdr - tpam
      dti = tpig - tpam
      ratio = ( hvap + cpm * dtb ) / ( chtd * dti )
      rhoc = dend * chtd * ( 1. + fmof * ratio )
      tmig = ( ( beta / hbar ) ** 2 ) * conw * rhoc

      return
      end
```

```
   subroutine DRYTIM( enu , theta , tau )

Given a Nusselt number ( enu , actually Biot number = h D / k )
and the dimensionless temperature rise required for the start
of surface drying ( theta ), returns the dimensionless time ( tau )
needed to achieve it.  The time is given multiplied by thermal
diffusivity and divided by radius squared.  Solution by binary search.

   data a / 0.7478556 / , b / 0.4653628 / , c / 0.1282064 /
   data p / 0.47047 /

   f( h ) = h * ( b - h * ( c - h ) ) - ( 1. - theta ) / a

   xl = 0.
   xh = 1.

   do n = 1 , 15
      xm = 0.5 * ( xl + xh )
      if( f( xm ) .LT. 0. ) then
         xl = xm
      else
         xh = xm
      end if
   end do

   x = ( 1. / xm - 1. ) / p
   tau = ( 0.5 * x / enu ) **2
   return
   end
```

```
      subroutine HEATX( u , d , dia , tf , ts , hfm , hbar , cond , en )

Given horizontal windspeed u at height d [top of fuelbed], cylindrical
fuel particle diameter dia, fire environment temperature tf, and mean
surface temperature, ts, subroutine returns film heat transfer coefficient
hfm and an "effective" film heat transfer coefficient including radiation
heat transfer, hbar.  Using the wood's thermal conductivity, cond, the
modified Nusselt number [ en ] used to estimate onset of surface drying
is returned as well.

      data g / 9.8 /
      data vis / 7.5 e - 05 /
      data a / 8.75 e - 03 /
      data b / 5.75 e - 05 /
      data rad / 5.67 e - 08 /
      data fmfac / 0.382 /
      data hradf / 0.5 /

      hfm = 0.
      if( dia .LE. b ) goto 10
      v = sqrt( u * u + 0.53 * g * d )
      re = v * dia / vis
      enuair = 0.344 * ( re ** 0.56 )
      conair = a + b * tf
      fac = sqrt( abs( tf - ts ) / dia )
      hfmin = fmfac * sqrt( fac )
      hfm = max( ( enuair * conair / dia ) , hfmin )
10    hrad = hradf * rad * ( tf + ts ) * ( tf * tf + ts * ts )
      hbar = hfm + hrad
      en = hbar * dia / cond
      return
      end
```

```
   function TEMPF( q , r , tamb )

Returns a fire environment temperature , TEMPF , given the fire intensity
q in kW / square meter , the ambient temperature tamb in Kelvins, and the
dimensionless mixing parameter r.

   data err / 1. e - 04 / , aa / 20. /
   term = r / ( aa * q )
   rlast = r
10 den = 1. + term * ( rlast + 1. ) * ( rlast * rlast + 1. )
   rnext = 0.5 * ( rlast + 1. + r / den )
   if( abs( rnext - rlast ) .LT. err ) then
       tempf = rnext * tamb
       return
   end if
   rlast = rnext
   goto 10
   end
```

```
  subroutine STEP( dt , mxstep , now , maxno , number , wo , alfa ,
 +                 dendry , fmois , cheat , condry , diam , tpig ,
 +                 tchar , xmat , tpamb , tpdry , fi , flit , fout ,
 +                 tdry , tign , tout , qcum , tcum , acum , qdot ,
 +                 ddot , wodot , work , u , d , r0 , dr , ch2o ,
 +                 ncalls , maxkl , tin , fint , fid )
```

Updates status of all fuel component pairs and returns a snapshot

Input parameters:

tin =	start of current time step
dt =	time step , sec
mxstep =	max dimension of historical sequences
now =	index marks end of time step
maxno =	max number of fuel components
number =	actual number of fuel components
wo =	current ovendry loading for the larger of each component pair, kg / sq m
alfa =	dry thermal diffusivity of component , sq m / s
dendry =	ovendry density of component , kg / cu m
fmois =	moisture fraction of component
cheat =	specific heat capacity of component , J / kg K
condry =	ovendry thermal conductivity , w / sq m K
diam =	current diameter of the larger of each fuel component pair , m
tpig =	ignition temperature (K) , by component
tchar =	end - pyrolysis temperature (K) , by component
xmat =	table of influence fractions between components
tpamb =	ambient temperature (K)
tpdry =	temperature (all components) start drying (K)
fi =	current fire intensity (site avg) , kW / sq m
work(k) =	factor of heat transfer rate hbar * (Tfire - Tchar) that yields ddot(k)
fint(k) =	correction to fi to compute local intensity that may be different due to k burning
fid =	fire intensity due to duff burning ... this is used to up the fire intensity for fuel pieces that are burning without interacting with others

plus the following constants and bookkeeping parameters
u , d , r0 , dr , ch20 , ncalls , maxkl

Parameters updated [input and output]

ncalls =	counter of calls to this routine ... = 0 on first call or reset cumulates after first call
flit =	fraction of each component currently alight
fout =	fraction of each component currently gone out
tdry =	time of drying start of the larger of each fuel component pair
tign =	ignition time for the larger of each fuel component pair
tout =	burnout time of larger component of pairs
qcum =	cumulative heat input to larger of pair , J / sq m

```
      tcum =          cumulative temp integral for qcum ( drying )
      acum =          heat pulse area for historical rate averaging
      qdot =          history ( post ignite ) of heat transfer rate
                      to the larger of component pair , W / sq m
      ddot =          diameter reduction rate , larger of pair , m / s
      wodot =         dry loading loss rate for larger of pair

Constant parameters

      u =             mean horizontal windspeed at top of fuelbed
      d =             fuelbed depth
      r0 =            minimum value of mixing parameter
      dr =            max - min value of mixing parameter
      ch2o =          specific heat capacity of water , J / kg K
      hvap =          heat of vaporization of water , J / kg

   real*4 wo( maxkl ) , alfa( maxno ) , dendry( maxno )
   real*4 fmois( maxno ) , cheat( maxno )
   real*4 condry( maxno ) , diam( maxkl ) , tpig( maxno )
   real*4 tchar( maxno ) , xmat( maxkl )
   real*4 tdry( maxkl ) , tign( maxkl )
   real*4 tout( maxkl ) , qcum( maxkl )
   real*4 tcum( maxkl ) , acum( maxkl )
   real*4 qdot( maxkl , mxstep ) , ddot( maxkl )
   real*4 wodot( maxkl ) , flit( maxno ) , fout( maxno )
   real*4 work( maxno ) , fint( maxno )
   real*4 rindef / 1.e+30 /

   logical flag

   loc( k , l ) = k * ( k + 1 ) / 2 + l

   ncalls = ncalls + 1
   tnow = tin
   tnext = tnow + dt
tifi = time when fire ignition phase ended ( at now = 1 )
   tifi = tnow - float( now - 1 ) * dt
   next = now + 1

   do k = 1 , number
      c = condry( k )
      do l = 0 , k
         kl = loc( k , l )
         tdun = tout( kl )

See if k of ( k , l ) pair burned out

         if( tnow .GE. tdun ) then
            ddot( kl ) = 0.
            wodot( kl ) = 0.
            goto 10
         end if
         if( tnext .GE. tdun ) then
            tgo = tdun - tnow
            ddot( kl ) = diam( kl ) / tgo
```

```
            wodot( kl ) = wo( kl ) / tgo
            wo( kl ) = 0.
            diam( kl ) = 0.
            goto 10
         end if

k has not yet burned out ... see if k of ( k , l ) pair is ignited

         tlit = tign( kl )
         if( tnow .GE. tlit ) then
            ts = tchar( k )
            if( l .EQ. 0 ) then
                r = r0 + 0.5 * dr
                gi = fi + fid
            end if
            if( ( l .NE. 0 ) .AND. ( l .NE. k ) ) then
                r = r0 + 0.5 * ( 1. + flit( l ) ) * dr
                gi = fi + fint( k ) + flit( l ) * fint( l )
            end if
            if( l .EQ. k ) then
                r = r0 +  0.5 * ( 1. + flit( k ) ) * dr
                gi = fi + flit( k ) * fint( k )
            end if
            tf = tempf( gi , r , tpamb )
            dia = diam( kl )
            call heatx( u , d , dia , tf , ts , hf , hb , c , e )
            qqq = hb * max( ( tf - ts ) , 0. )
            tst = max( tlit , tifi )
            nspan = max( 1 , nint( ( tnext - tst ) / dt ) )
            if( nspan .LE. mxstep ) qdot( kl , nspan ) = qqq
            if( nspan .GT. mxstep ) then
               do mu = 2 , mxstep
                  qdot( kl , mu - 1 ) = qdot( kl , mu )
               end do
               qdot( kl , mxstep ) = qqq
            end if
            aint = ( c / hb ) ** 2
            acum( kl ) = acum( kl ) + aint * dt
            tav1 = tnext - tlit
            tav2 = acum( kl ) / alfa( k )
            tav3 = ( ( dia / 4. ) ** 2 ) / alfa( k )
            tavg = min( tav1 , tav2 , tav3 )
            index = 1 + min( nspan , mxstep )
            qdsum = 0.
            tspan = 0.
            deltim = dt
01          index = index - 1
            if( index .EQ. 1 ) deltim = tnext - tspan - tlit
            if( ( tspan + deltim ) .GE. tavg ) deltim = tavg - tspan
            qdsum = qdsum + qdot( kl , index ) * deltim
            tspan = tspan + deltim
            if( ( tspan .LT. tavg ) .AND. ( index .GT. 1 ) ) goto 01
            qdavg = max( qdsum / tspan , 0. )
            ddot( kl ) = qdavg * work( k )
            dnext = max( 0. , dia - dt * ddot( kl ) )
```

```
            wnext = wo( kl ) * ( ( dnext / dia ) ** 2 )
            if( ( dnext .EQ. 0. ) .AND. ( ddot( kl ) .GT. 0. ) ) then
               tout( kl ) = tnow + dia / ddot( kl )
            end if
            if( ( dnext .GT. 0. ) .AND. ( dnext .LT. dia ) ) then
               rate = dia / ( dia - dnext )
               tout( kl ) = tnow + rate * dt
            end if
            if( qdavg .LE. 20. ) tout( kl ) = 0.5*( tnow + tnext )
            ddt = min( dt , ( tout( kl ) - tnow ) )
            wodot( kl ) = ( wo( kl ) - wnext ) / ddt
            diam( kl ) = dnext
            wo( kl ) = wnext
            goto 10
         end if

 See if k of ( k , l ) has reached outer surface drying stage yet

         dryt = tdry( kl )
         if( ( tnow .GE. dryt ) .AND. ( tnow .LT. tlit ) ) then
            if( l .EQ. 0 ) then
               r = r0
               gi = fi + fid
            end if
            if( l .EQ. k ) then
               r = r0
               gi = fi
            end if
            if( ( l .NE. 0 ) .AND. ( l .NE. k ) ) then
               r = r0 + 0.5 * flit( l ) * dr
               gi = fi + flit( l ) * fint( l )
            end if
            tf = tempf( gi , r , tpamb )
            ts = tpamb
            dia = diam( kl )
            call heatx( u , d , dia , tf , ts , hf , hb , c , e )
            dtemp = max( 0., ( tf - ts ) )
            dqdt = hb * dtemp
            qcum( kl ) = qcum( kl ) + dqdt * dt
            tcum( kl ) = tcum( kl ) + dtemp * dt
            dteff = tcum( kl ) / ( tnext - dryt )
            heff = qcum( kl ) / tcum( kl )
            tfe = ts + dteff
            dtlite = rindef
            if( tfe .LE. ( tpig( k ) + 10. ) ) goto 02
            call TIGNIT( tpamb , tpdry , tpig( k ) , tfe ,
     +           condry( k ) , cheat( k ) , fmois( k ) , dendry( k ) ,
     +           heff , dtlite )
02          tign( kl ) = 0.5 * ( dryt + dtlite )

 If k will ignite before time step over , must interpolate

            if( tnext .GT. tign( kl ) ) then
               ts = tchar( k )
               call heatx( u , d , dia , tf , ts , hf , hb , c , e )
```

```
            qdot( kl , 1 ) = hb * max( ( tf - ts ) , 0. )
            qd = qdot( kl , 1 )
            ddot( kl ) = qd * work( k )
            delt = tnext - tign( kl )
            dnext = max( 0. , dia - delt * ddot( kl ) )
            wnext = wo( kl ) * ( ( dnext / dia ) ** 2 )
            if( dnext .EQ. 0. ) tout( kl ) = tnow + dia / ddot(kl)
            if( ( dnext .GT. 0. ) .AND. ( dnext .LT. dia ) ) then
               rate = dia / ( dia - dnext )
               tout( kl ) = tnow + rate * dt
            end if
            if( tout( kl ) .GT. tnow ) then
               ddt = min( dt , ( tout( kl ) - tnow ) )
               wodot( kl ) = ( wo( kl ) - wnext ) / ddt
            else
               wodot( kl ) = 0.
            end if
            diam( kl ) = dnext
            wo( kl ) = wnext
         end if
         goto 10
      end if
```

If k of (k , l) still coming up to drying temperature , accumulate heat input and driving temperature difference , predict drying start

```
      if( tnow .LT. dryt ) then
         factor = fmois( k ) * dendry( k )
         conwet = condry( k ) + 4.27e-04 * factor
         if( l .EQ. 0 ) then
            r = r0
            gi = fi + fid
         end if
         if( l .EQ. k ) then
            r = r0
            gi = fi
         end if
         if( ( l .NE. 0 ) .AND. ( l .NE. k ) ) then
            r = r0 + 0.5 * flit( l ) * dr
            gi = fi + flit( l ) * fint( l )
         end if
         tf = tempf( gi , r , tpamb )
         if( tf .LE. ( tpdry + 10. ) ) goto 10
         dia = diam( kl )
         ts = 0.5*( tpamb + tpdry )
         call heatx( u , d , dia , tf , ts , hf , hb , c , e )
         dtcum = max( ( tf - ts ) * dt , 0. )
         tcum( kl ) = tcum( kl ) + dtcum
         qcum( kl ) = qcum( kl ) + hb * dtcum
         he = qcum( kl ) / tcum( kl )
         dtef = tcum( kl ) / tnext
         thd = ( tpdry - tpamb ) / dtef
         if( thd .GT. 0.9 )  goto 10
         biot = he * dia / conwet
         call DRYTIM( biot , thd , dryt )
```

```
            cpwet = cheat( k ) + ch2o * fmois( k )
            fac = ( ( 0.5 * dia ) ** 2 ) / conwet
            fac = fac * cpwet * dendry( k )
            tdry( kl ) = fac * dryt
            if( tdry( kl ) .LT. tnext ) then
               ts = tpdry
               call heatx( u , d , dia , tf , ts , hf , hb , c , e )
               dqdt = hb * ( tf - ts )
               delt = tnext - tdry( kl )
               qcum( kl ) = dqdt * delt
               tcum( kl ) = ( tf - ts ) * delt
               tbar = 0.5 * ( tpdry + tpig( k ) )

 See if ignition to occur before time step complete

               if( tf .LE. ( tpig( k ) + 10. ) ) goto 10
               call TIGNIT( tpamb , tpdry , tpig( k ) , tf ,
     +             condry( k ) , cheat( k ) , fmois( k ) , dendry( k ) ,
     +             hb , dtlite )
               tign( kl ) = 0.5 * ( tdry( kl ) + dtlite )
               if( tnext .GT. tign( kl ) ) then
                  ts = tchar( k )
                  qdot( kl , 1 ) = hb * max( ( tf - ts ) , 0. )
               end if
            end if
         end if
10       continue
      end do
   end do

 Update fractions ignited and burned out , to apply at next step start

   do k = 1 , number
      flit( k ) = 0.
      fout( k ) = 0.
      do l = 0 , k
         kl = loc( k , l )
         flag = ( tnext .GE. tign( kl ) )
         if( flag .AND. ( tnext .LE. tout( kl ) ) ) then
            flit( k ) = flit( k ) + xmat( kl )
         end if
         if( tnext .GT. tout( kl ) ) then
            fout( k ) = fout( k ) + xmat( kl )
         end if
      end do
   end do

   return
   end
```

```
      subroutine SUMMARY( outfil , number , maxno , maxkl, parts , nun ,
     +tis , ak , wdry , fmois , sigma , tign , tout , xmat , wo , diam )

      real*4 wdry( maxno ) , fmois( maxno ) , sigma( maxno )
      real*4 tign( maxkl ) , tout( maxkl ) , wo( maxkl )
      real*4 xmat( maxkl ) , diam( maxkl )
      character*12 parts( maxno ) , outfil , name , none , nuname
      character*3 stat
      logical v

      loc( k , l ) = k * ( k + 1 ) / 2 + l

      none = 'no companion'
      nuname = outfil
      stat = 'NEW'

01    write( * , 05 )
05    format(' Full summary = 1 , recap only = 0    ',$ )
      read( * , * ) ido
      v = ( ido .EQ. 1 )
      if( ( ido .NE. 0 ) .AND. ( .NOT. v ) ) goto 01

10    open( nun , file=nuname , status=stat , form='FORMATTED' , err=99)
      write( nun , 15 ) ak , tis
15    format(5x'Area factor =' , f6.2 , '  End time , s =' , e12.4 )
20    format(/5x,a12,3x'Load , mois , diam =  '3e10.3/
     +5x'  Companion  fraction   Ignition   burnout    load rem     diam')
21    format(6x'Component    load @ 0    Ignition    burnout    load rem')
30    format(5x,a12,e9.3,e11.3,e10.3,e11.3,e9.3)
      if( .NOT. v ) write( nun , 21 )
35    do m = 1 , number
         name = parts( m )
         win = wdry( m )
         fmi = fmois( m )
         dim = 4. / sigma( m )
         if( v ) write( nun , 20 ) name , win , fmi , dim
         name = none
         rem = 0.
         ts = 1.e+31
         tf = 0.
         do n = 0 , m
            mn = loc( m , n )
            fr = xmat( mn )
            ti = tign( mn )
            ts = min( ti , ts )
            to = tout( mn )
            tf = max( to , tf )
            wd = wo( mn )
            rem = rem + wd
            di = diam( mn )
            if( v ) write( nun , 30 ) name , fr , ti , to , wd , di
            if( n .LT. m ) name = parts( ( n + 1 ) )
         end do
         name = parts( m )
         if( .NOT. v ) write( nun , 40 ) name , win , ts , tf , rem
```

```
      end do
 40   format(5x,a12,e9.3,e11.3,e10.3,e11.3)
      return

 99   write(*,100) nuname , stat
100   format('  Error on open:  'a12'  with status = 'a3/
     +       '  0 = abort now'/
     +       '  1 = overwrite existing file'/
     +       '  2 = enter new file name to create   ',$)
      read(*,*) in
      if( in .EQ. 0 ) stop' Abort'
      if( in .EQ. 1 ) then
          stat = 'OLD'
          goto 10
      end if
      if( in .NE. 2 ) goto 99
      write(*,*)' Enter new file name [max 12 characters]'
      read(*,101) nuname
101   format(a12)
      stat = 'NEW'
      open( nun , file=nuname , status=stat , form='FORMATTED' , err=99)
      goto 35

      end
```